

Project: Scheduling Web-App Development and Security Testing

Author: Huzaifa Patel

Project Report

Abstract

Secure Scheduling System is a scheduling web application that addresses the flaws in existing scheduling solutions, most notably lack of security features and high costs. Research was conducted to identify the vulnerabilities in the scheduling systems used by small businesses, and I have found a lack of protection for sensitive data and poor error management which can leave user data exposed to cyber threats. (OWASP, 2024)

This solution is designed for small businesses and combines strong web security features, including compliance with the OWASP Top 10 and the Cyber Essentials scheme. The key features include secure access controls, password hashing for user authentication, efficient session management and a user-friendly calendar system with CRUD operations. The backend is built on the PHP-based Laravel framework. The MVC design pattern is used to securely integrate the MySQL database, resulting in better scalability and maintenance.

The project was designed, developed, and continuously improved using the Agile Development Methodology and feedback from both project supervisors and end users. To prioritise security, the application was regularly tested throughout the development process. This included security testing by manually targeting the application using penetration testing tools on Kali Linux such as Nmap and Metasploit. Iterative testing ensured that the project complies with industry security standards to provide customers with a safe, user-friendly scheduling tool. The evaluation concludes that the project requirements were met as this web application addresses the gaps in security and accessibility that were found in existing solutions.

Table of Contents

Table of Contents.....	3
Chapter 1: Introduction.....	5
1.1 Motivation	5
1.2 Aim	6
1.3 Objectives	6
1.4 Adopted Approach.....	6
1.5 Overview of the report.....	7
Chapter 4: Requirements, Specification and Design	7
4.1 Introduction	7
4.2 Requirements.....	7
4.2.1 Functional Requirements	8
4.2.2 Non-functional Requirements.....	8
4.3 Specification and Design.....	9
4.3.1 Wireframes.....	10
4.3.2 Applications, Tools and Programming Languages.....	12
4.3.3 Database Design.....	13
4.3.4 MVC Architecture Design	14
Chapter 5: Development and Implementation	15
5.1 Introduction	15
5.2 Database.....	15
5.3 CRUD Operations.....	16
5.3.1 CRUD Backend.....	16
5.3.2 CRUD Frontend	17
5.3.3 Problems encountered.....	19
5.4 User authorization and authentication	20
5.4.1 Homepage	20
5.4.2 Login and Register.....	20
5.4.3 Problems encountered.....	22
5.5 Account management	22
5.5.1 Password Reset	24

5.5.2 Problems encountered.....	26
5.6 Calendar Displays.....	26
5.6.1 User Calendar.....	27
5.6.2 Admin Calendar.....	28
5.6.3 Problems encountered.....	28
5.7 Custom error handling.....	29
5.8 Middleware.....	29
5.8.1 Problems encountered.....	30
5.9 Summary.....	30
Chapter 6: Testing and Analysis.....	31
6.1 Introduction	31
6.2 Regression testing.....	31
6.3 Usability testing.....	33
6.4 Penetration testing	33
6.4.1 Virtual Machine Setup	33
6.4.2 Brute force attack.....	34
6.4.3 CSRF Vulnerability.....	35
6.4.4 SQL Injection.....	36
6.4.5 SQL Injection – Problems encountered	36
6.5 Web testing.....	37
6.5.1 Introduction.....	37
6.5.2 Markup and CSS validation.....	38
6.5.3 Independent browser and platform functional testing.....	38
6.6 Analysis of the testing results.....	40
Chapter 7: Lessons learnt.....	41
Chapter 8: Conclusions	42
8.1 Project Overview.....	42
8.3 Legal, social, ethical and professional issues and Acknowledgements.....	42
8.3.1 Acknowledgements.....	42
Bibliography	43

Chapter 1: Introduction

1.1 Motivation

Online scheduling systems are essential to the day-to-day operations of businesses across all industries. They allow employees to manage their daily tasks with a straightforward booking system for shifts and workspaces. Scheduling systems also improve a business's operations by optimizing the management and coordination of human resources and office spaces.

This demand for these systems is increasing as full-time and hybrid remote work is becoming more popular and easy access to booking work shifts is more sought after.

The leading motivation for this project is my personal experiences with my previous employer's shift scheduling system, which had an outdated UI and often exposed vulnerabilities to users. The website lacked standard security features, and one security flaw was detailed internal error messages being displayed after a login attempt, regardless of whether the login was successful or not. The error messages were detailed and revealed information about the database queries and programming language used, information that presents a significant attack vector.

Attackers can conduct passive reconnaissance using open-source intelligence (OSINT) techniques to find this information and use it to attack vulnerabilities in the application. This issue highlights how little importance is given to securing these systems, especially from small to medium sized businesses, despite these attack vectors posing a great threat to their overall cybersecurity structure. Scheduling systems are used by employees daily therefore, it is critical that they must always be secure.

Familiar and well-established solutions exist for scheduling systems, however they are often very expensive which can make them unaffordable for small businesses. The databases for these platforms are shared across multiple businesses which makes them a high value target for cyberattacks. Some solutions are mobile apps which are designed only for mobile use which restricts accessibility for some users. Creating a web application would cater to more users and offer a custom-built solution that fits their needs.

This project will strengthen my professional development in cyber security with secure web development, penetration testing and database security. Following the OWASP Top 10 and Cyber Essentials standard during the development process will develop my secure coding and information security management skills.

1.2 Aim

This project aims to create a secure web application that will help businesses with their scheduling needs. The development will follow the Cyber Essentials standard and OWASP Top 10 to ensure industry standard development practices and security.

1.3 Objectives

The objectives have been reviewed and updated since the first version in the Project Proposal.

The following list contains the main objectives for this project:

- Objective 1: To research the foundations of web security including the OWASP top 10, the common cyberattack vectors and the security responses.
- Objective 2: To research products related to scheduling tools to improve the features currently available, understand how to improve accessibility, then design the user interface to be responsive and accessible.
- Objective 3: To research database management systems.
- Objective 4: To research suitable programming languages, frameworks and libraries (HTML, React, Django, Angular, *Laravel*) and software that can be used to develop the web application – focusing on the security benefits.
- Objective 5: To arrange approved web server / hosting facilities for the project by February 2025.
- Objective 6: To develop the login and user profile system with authorization and principle of least privilege.
- Objective 7: To develop the method of connecting the web application to the database, ensuring protection from common attacks such as SQL injection and cross-site scripting (XSS).
- Objective 8: To develop the create, read, update and delete (CRUD) functionality for scheduling events.
- Objective 9: To perform security testing using penetration testing tools on Kali Linux to evaluate system vulnerabilities and document the results.
- Objective 10: To evaluate the application's performance, accessibility and security.

Additionally, I have set the following optional objectives for this project to be completed once the main objectives have been completed:

- To research other clients for this web application (Hospitals, Libraries, Educational Institutions, Shared Offices).
- Further develop the scheduling system to allow for multiple use cases.
- Develop additional features such as email/SMS notifications, 2fa for logins, and mobile support.

1.4 Adopted Approach

I started this project by carrying out research to form a basis for my project. I began by researching the core web security features using the OWASP Top 10 to understand the most critical aspects of web security. The UK

Cyber Essentials help to protect against almost all cyber threats and will be reviewed for controls that I plan to implement in my project. (Gov.uk, 2024)

I have selected Agile as the methodology for my project. Agile is about sustainable development and promotes a steady pace of development. (Risener, 2022) The Agile Manifesto focuses on simplicity, whereas other methodologies such as Scrum have stricter definitions and development processes. Having compared different methodologies, I have found Agile to be the most suitable methodology for this project. The flexible development process and incremental improvements in the Agile methodology work better with my goals and help to keep the development focused on the project objectives.

1.5 Overview of the report

The purpose of this dissertation is to detail the steps taken throughout this project. The first part of this report consists of the first 3 sections, where the literature review, chosen methodology, and Requirements, Specification and Design are discussed. The second part of this report discusses the development of the web application and the different types of testing that was performed. The report concludes with a critical evaluation of the project.

Chapter 4: Requirements, Specification and Design

4.1 Introduction

This chapter will outline the main requirements and designs for this project. As the project followed the agile methodology, the first few sprints were dedicated to drafting designs and creating the main objectives following inspiration from similar products identified during the literature review.

4.2 Requirements

The requirements were initially created based on my motivation for this project, which was my personal experiences with business scheduling systems. Through this experience I had found gaps in the security and accessibility of these systems. The incremental agile development methodology allows for additional requirements to be introduced throughout the development process. Weekly meetings were conducted with supervisors and feedback was taken from customers to review the vision for the project and to initially set the main requirements. Security and user accessibility were the main requirements for the project and user stories were created to introduce specific requirements and represent how these could be implemented to benefit customers. These requirements were created by reviewing articles and academic papers relating to web security

and scheduling systems, and then categorised into non-functional requirements, functional requirements and security requirements.

User stories have been used to outline how specific features will benefit customers and encourage myself to think critically and creatively about how to best solve for the end goal of this project. The requirements have been implemented in the user stories below.

4.2.1 Functional Requirements

Functional requirements define the core features and functions of the application. These requirements were created and updated through the weekly feedback sessions with the project supervisors and customers. They include the necessary back-end features such as CRUD operations and the front-end features to accurately display events. In short, these are the main requirements for the scheduling app.

Story no.	Requirement
1	As a user, I want the web application to be responsive and load quickly.
2	As a user, I want the interface to be accessible with readable fonts.
3	As a user, I want the web application to be responsive and fit multiple screen sizes.
4	As a user, I want my password information to be stored securely with encryption.
5	As a user, I want the web application to validate all inputs with protection against malicious inputs.
6	As a user, I want the web application to have proper authentication and session management so that unauthorized users cannot access sensitive information.
7	As a user, I want clear error messages that do not include insights into the system configuration to minimise security risks.
8	As a user, write and update functionality should only be available to admin users.
9	As a user, read functionality should only be available to users who are logged in.
10	As an admin, I want to be able to create and delete events.
11	As an admin, I want to be notified when I submit invalid data so that I can correct it.
12	As a user, I want the navigation bar to be uniform across all webpages and easily accessible with relevant webpage titles.
13	As a user, I want the colours to be readable whilst being accessible to all users including the colourblind.

Table 1: Functional requirements – User Stories

4.2.2 Non-functional Requirements

The non-functional requirements are the attributes of the system which extend beyond the core functionality to ensure the web application meets user expectations. This includes accessibility, design and performance as well as the security features. Following the literature review, I have decided to implement the Cyber Essentials and OWASP Top 10 as a basis for this web application and began by representing the recommendations as user stories.

Story no.	Requirement
1	As a user, I want the web application design to be similar to other systems I have used to increase usability and minimise the learning curve.
2	As a user, I want to view all my events in a calendar format so that I can easily track my daily schedule.
3	As a user, I want to view all my events in a list format so that I can easily track my daily schedule.
4	As an admin, I want the ability to customise my event's title, date, time, and location so that I can manage my/my team's schedule efficiently.
5	As an admin, I want to be able to edit existing events so that I can update details as and when needed.
6	As an admin, I want all users to register for an account or login before using the scheduling system.
7	As a user, I want secure password recovery/reset functionality.
8	As a user, I want to be able to update my login details.
9	As a user, I want to be able to access the web application from multiple browsers.
10	As an admin, I want the solution to be easily scalable to support the growth of my business.

Table 2: Non-functional requirements - User Stories

4.3 Specification and Design

This subsection outlines the different technologies that have been used in this project, with justifications for why they were selected. The product design including the user interface wireframes for the website follows the minimalist design language used by similar products to achieve non-functional requirement 1.

4.3.1 Wireframes

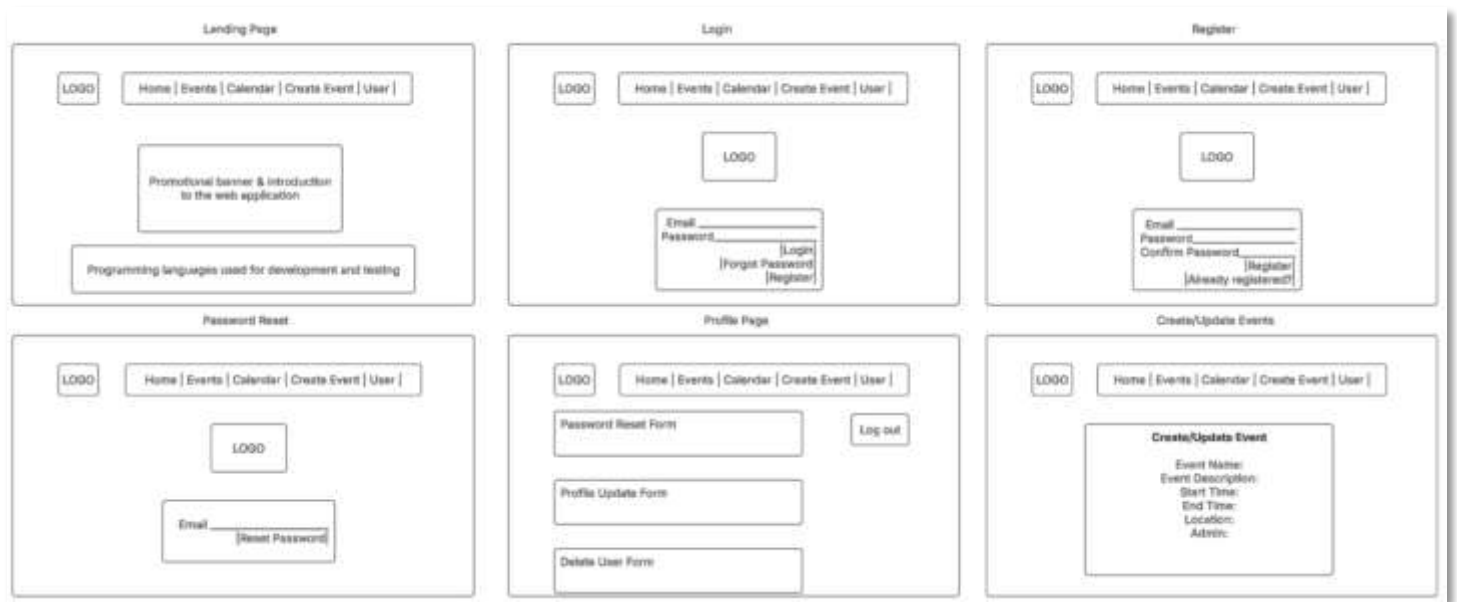


Figure 5: Website Wireframes

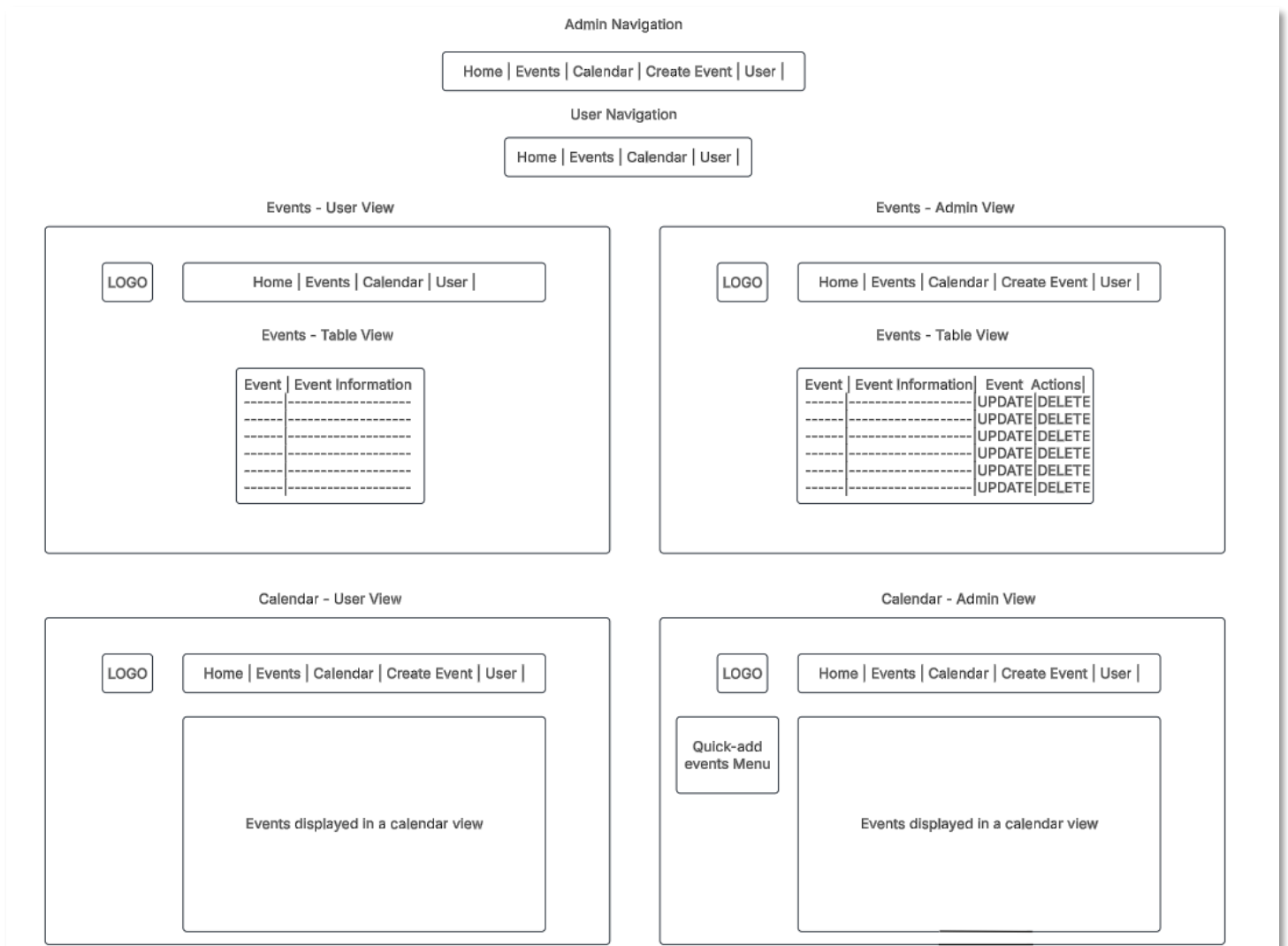


Figure 6: Website Wireframes

The literature review revealed that wireframes play an important role in web design. It is important that all the requirements, including the security requirements are kept in mind during this first step of the development process. The functional requirements 12 and 13 have been addressed by implementing accessible design choices throughout the website such as the uniform navigation bar which improves the ease of use of the site. (Flavian, Gurrea and Orus, 2009) A uniform user interface design is implemented across all webpages and elements such as a large calendar view are adequately sized to promote accessibility for the visually impaired. The non-functional requirements 1, 2, 3 and 9 have been addressed with separate webpages to display events in both a calendar and list view with a user interface that is similar to the industry standard products found during the research phase of the literature review.

Security has been a fundamental requirement throughout this project and the aspects related to the privacy and security have been considered in every time and place of the website. (Flavian, Gurrea and Orus, 2009) These wireframes were designed to include separate navigation bars for logged in and logged out users and admins which will complement the robust session management during development. Users will be forced to login

before accessing any webpage other than the landing webpage to ensure users do not access unauthorised information.

These wireframes comply with all the requirements for the project, and this process of planning the development of this project around the requirements will ensure that the final product is a secure and accessible scheduling system.

4.3.2 Applications, Tools and Programming Languages

This section highlights the tools and applications that have been used during the development of this web application. Also included are the programming languages used as well as the Laravel PHP framework.

4.3.2.1 Web Server Solutions: XAMPP and Poseidon

The Apache distribution XAMPP was used to host the local web application during development and testing. Poseidon also uses a personal computer for hosting; however, it allows for live hosting meaning users can also access the web application. The Poseidon web hosting services were unavailable during the development of the web application, so arrangements were made to transfer the application from XAMPP once the minimum viable product had been achieved.

XAMPP is the perfect solution as this web application utilises the Apache web server and MySQL for the database. PHP is the main programming language used for this web app and XAMPP is the most popular PHP development environment. (Apache Friends, 2022)

When hosted on XAMPP, the web application is limited to localhost and not exposed to the internet which makes it suitable for hosting during security testing. This is important as only this web application will be affected if the server crashes or becomes corrupted during security testing. Penetration testing a live web application can result in crashing the hosting platform e.g. GoDaddy and affecting the other services hosted online which has both ethical concerns and legal consequences. (SoftwareSecured, n.d.)

4.3.2.2 MySQL Database

As mentioned in Section 4.2.2.1, the MySQL integration in XAMPP was used as the database management system for the backend of the web application. It was the main data storage solution, managing and structuring all data relating to users, events, sessions and the access control mechanisms used for security purposes. The fast and reliable data retrieval, as well as compatibility with the chosen Laravel framework and all operating systems makes this application easily scalable to support the growth of my customers' businesses. It was also used to ensure robust security by storing hashed passwords and integrating Laravel's query builder to protect against SQL injection.

4.3.2.3 Laravel (PHP), HTML, CSS, Tailwind and JavaScript

PHP was the main programming language for both the backend and frontend and was used within the Laravel framework to handle interactions between the components within the MVC architecture [Section 4.2.4]. This

included routing, form processing, user authentication and the database interactions. HTML was used to structure some of the user interfaces where PHP was not used.

The Laravel framework is built on PHP and includes secure coding practices throughout the coding process for this project. It provided the structure for routing with middleware authentication as well as form validation and CSRF protection. This ensured the core functionality of the web application was built on the project requirements for security and scalability.

CSS was used to style most of the HTML elements, while Tailwind was used to style some of the non-repeated elements. JavaScript was used to implement Full Calendar for the calendar user interface.

This stack of programming languages was used to achieve the project’s requirements for accessibility, responsiveness and security.

4.3.2.4 Lucid.app

Lucid.app includes both the Lucid chart and Lucid spark software which were used to design the diagrams for this project. This included the wireframes for the user interface and the MVC Architecture in Section 4.2.4.

4.3.3 Database Design

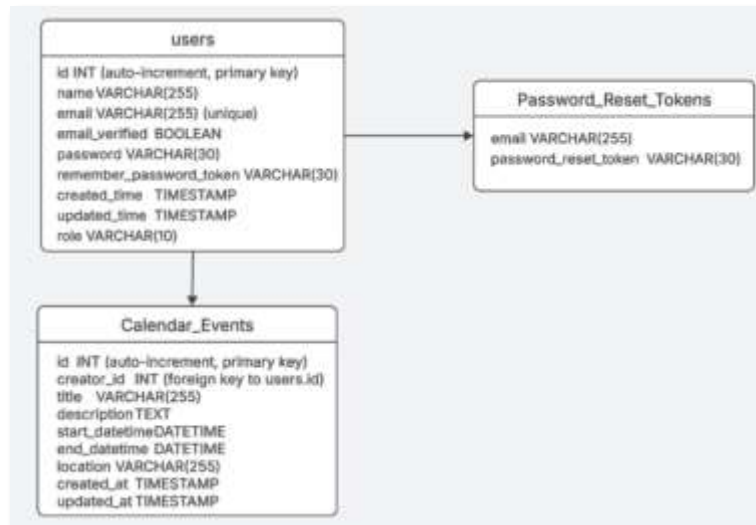


Figure 7: Database design

The database was designed to meet the functional, non-functional, and security requirements for the project most notably password hashing and role-based access control.

The users table includes the information about each user. It includes a hashed password to enhance security, remember_password_token for better accessibility and a role column used to validate users for role-based access control. Considerations were made to include the password_reset table for users who need to reset their credentials allowing for easy password updates to improve accessibility. This table includes the email row which makes sure the email is associated with an account before sending the password reset email. The

calendar_events table holds the information about events that have been created and is linked to the users table through creator_id which ensures role-based access as only admin users should be able to create events.

Timestamps and foreign keys are used for data integrity and to track changes in the records, making sure that the system is consistently reliable. This database design allows for secure CRUD operations with role-based access controls and is easily scalable. It also meets the non-functional requirements by promoting accessibility for users who may have trouble remembering passwords.

4.3.4 MVC Architecture Design

The MVC design pattern was used to develop the logic for bridging the backend of the web application to the user interface. MVC separates the application into Model, View and Controller.

The model is the classes that contain the logical operations and database interactions/queries.

The view is the classes responsible for the user interface elements.

The controller is the class that is handling input and controlling interactions between the Model and View.

[Section 3.6]

To better adapt the MVC architecture for this application, the Route was implemented into the architecture as shown below.

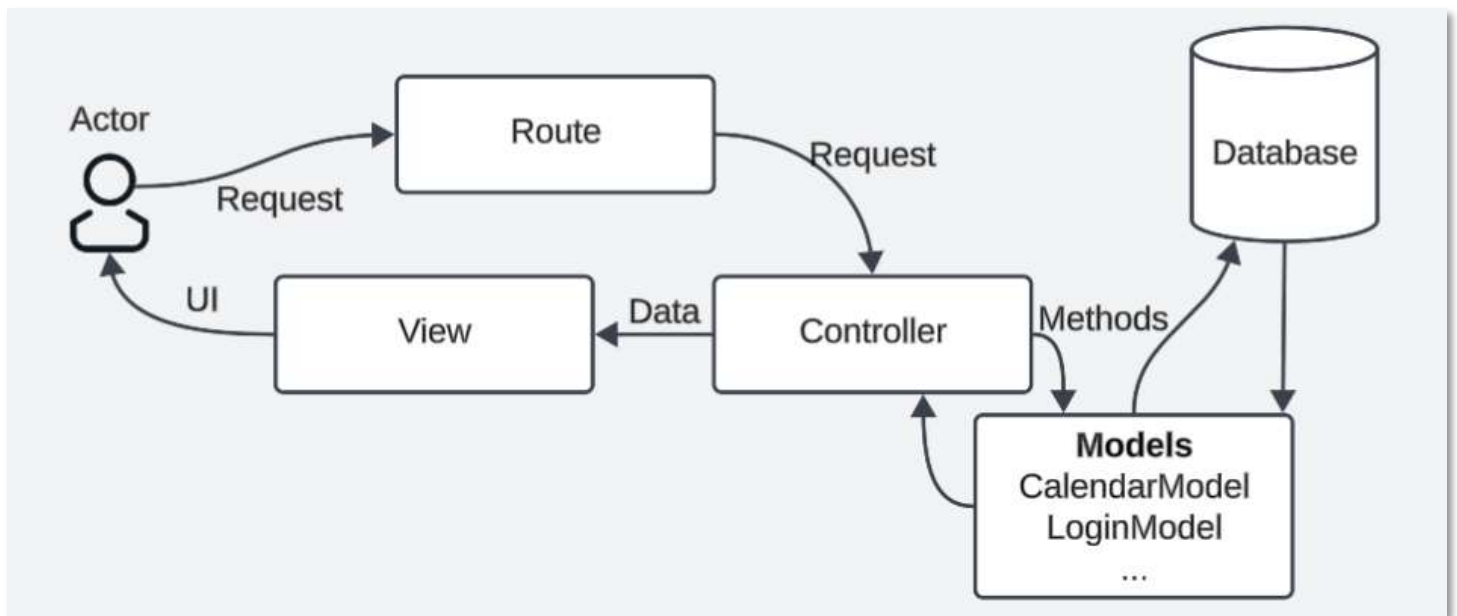


Figure 8: MVC Architecture Design

The actor represents the end-users interacting with the web app. All requests go through the Route which directs them to the appropriate controller. This layer includes authentication such as middleware and CSRF tokens to enforce security and validate the request before reaching the controller. The controller processes the request with input sanitisation after it is validated by the Route, then interacting with the Models to receive the necessary information before sending this to the View. The models handle the CRUD operations from the database, with each model interacting with a different table to ensure role-based access. The Database stores the

raw data, which is transferred through both the models and controller to validate the data before it is sent to the View. The purpose of the View is to present the data to the actor providing not only beauty and appeal, but also high levels of usability for all types of users. (Flavian, Gurrea and Orus, 2009)

This MVC architecture ensures a clear separation between the users, data access logic and presentation of the data with each layer focusing on its responsibility. Requests are constantly validated throughout the process starting from input sensitisation, middleware authentication, then CSRF token validation at the Models all before the data has been requested from the database. Security has been at the core of designing this MVC architecture, aiming to keep the web application functioning smoothly and protecting business from cyber vandalism, data theft, unethical competition, and other negative consequences. (Cloudflare, 2024)

Chapter 5: Development and Implementation

5.1 Introduction

This chapter outlines the development process for this project and how the requirements and designs from chapter 4 have been implemented into the final product. The development process consisted of sprints, following the Agile methodology. As mentioned in Section 2, the flexibility of this methodology allows me to focus on the project objectives and constantly look back at my work to see if there is any room for improvement, allowing me to work to the best of my ability with maximum productivity.

The security functionality across the webapp uses Laravel's csrf tokens, authorisation middleware, input validation and secure routing to meet the project's security requirements. The Blade engine is used, for example all webpages are in '.blade.php' format to help prevent XSS (Cross-Site Scripting) attacks. Laravel documentation was used to learn and implement these features (Laravel, 2025) Web security was the main goal during the development and implementation of this web app.

5.2 Database

The development process started with creating the database. The initial requirements were gathered through stakeholder meetings with both the project supervisors and a potential client. This allowed me to refine the database requirements after each weekly sprint and identify the core entities such as users, events, and password resets. The priority was to build the foundation blocks for the CRUD operations and ensure security features such as role-based access controls would be present. MySQL was chosen as the database management system for its compatibility with Laravel and the database schema was designed to support user data, password resets, and calendar events.

Primary and foreign key relationships were used to maintain integrity between linked tables and ensured users could only manipulate their own data. This addressed A1: Broken Access Control from OWASP. (OWASP, 2024) Passwords were hashed before storage in line with A3: Sensitive Data Exposure. Prepared statements and

Laravel's Eloquent ORM were used as a crucial defence against SQL injection and meet Cyber Essentials' control for secure configuration. (NCSC, 2023)

The database was created by following the database design in chapter 4. Database migrations were implemented using Laravel's built-in migration system to allow secure connections from the PHPStorm IDE.

5.3 CRUD Operations

Following the development of the database, the next goal was creating the functionality of the Create, Read, Update, and Delete operations for both users and admins.

5.3.1 CRUD Backend

The backend logic for CRUD (Create, Read, Update, Delete) operations was developed using the Laravel framework and following the MVC (Model-View-Controller) architecture.

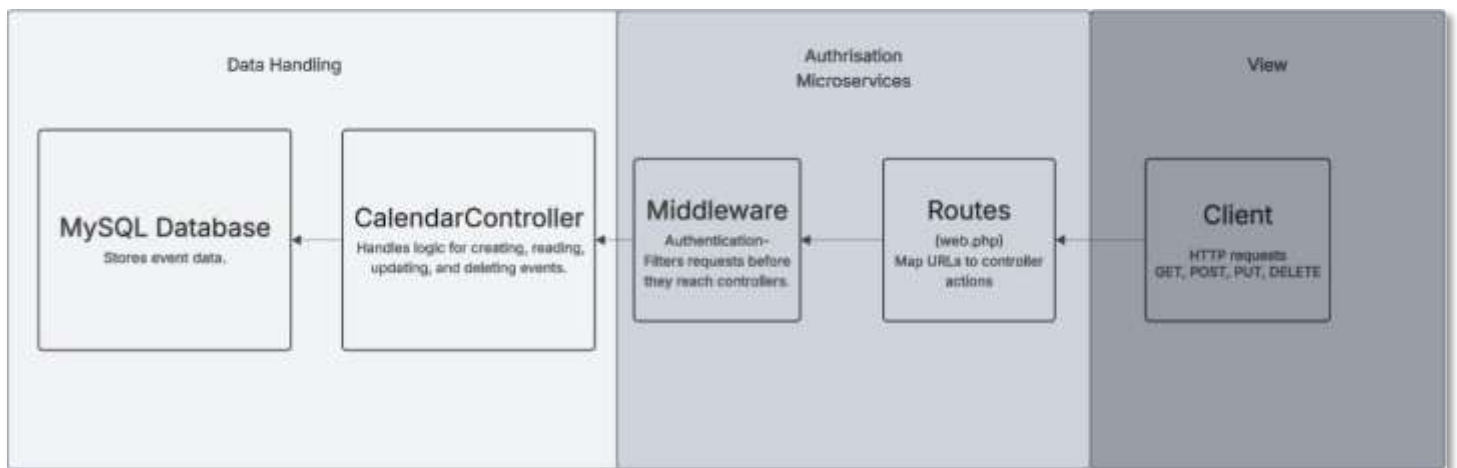


Figure 9: Backend diagram

The CalendarModel was created to represent calendar event entries in the database.

```
1 class CalendarModel extends Model
2 {
3     protected $table = 'calendar_events'; // database table name
4     // messages
5     protected $fillable = [
6         'user_id',
7         'title',
8         'description',
9         'start_datetime',
10        'end_datetime',
11        'location',
12    ];
13 }
14 public function user()
15 {
16     return $this->belongsTo('User', 'user_id', 'id');
17 }
```

Figure 10: Snippet from the CalendarModel.

Each CRUD operation was handled through the methods in the CalendarController.

Create: The store() method receives the validated form data and uses CalendarModel::create() to create a new event record.

Read: The index() and calendar() methods retrieve all events. These events are passed to the Laravel Blade views (UI pages) to be displayed to the user.

Update: The edit() method takes the event ID and loads the event's information whilst redirecting the user to the update webpage. Once loaded, the update webpage is auto filled with the existing information, while the update() method applies the validated changes to the corresponding record in the database.

Delete: The destroy() method uses the event ID to locate the event and delete the column from the database. The user is redirected to the same webpage, effectively reloading the webpage to show the changes.

```
class CalendarController extends Controller
{
    public function index()
    {
        $user = Auth::user();
        if ($user->role !== 'admin') {
            return redirect()->route('calendar.userindex')->with('error', 'You are not authorized to access');
        }
        $layout = 'layouts.main'; // Only display Admin layout if user is an admin
        $events = CalendarModel::all();

        return view('calendar.index', compact('var_name', 'events', 'var_names', 'layout'));
    }
}

// Usage
public function userindex()
{
    $user = Auth::user();
    // Redirect admins to the admin calendar page
    if ($user->role === 'admin') {
        return redirect()->route('calendar.index');
    }

    $layout = 'layouts.user'; // User layout
}
```

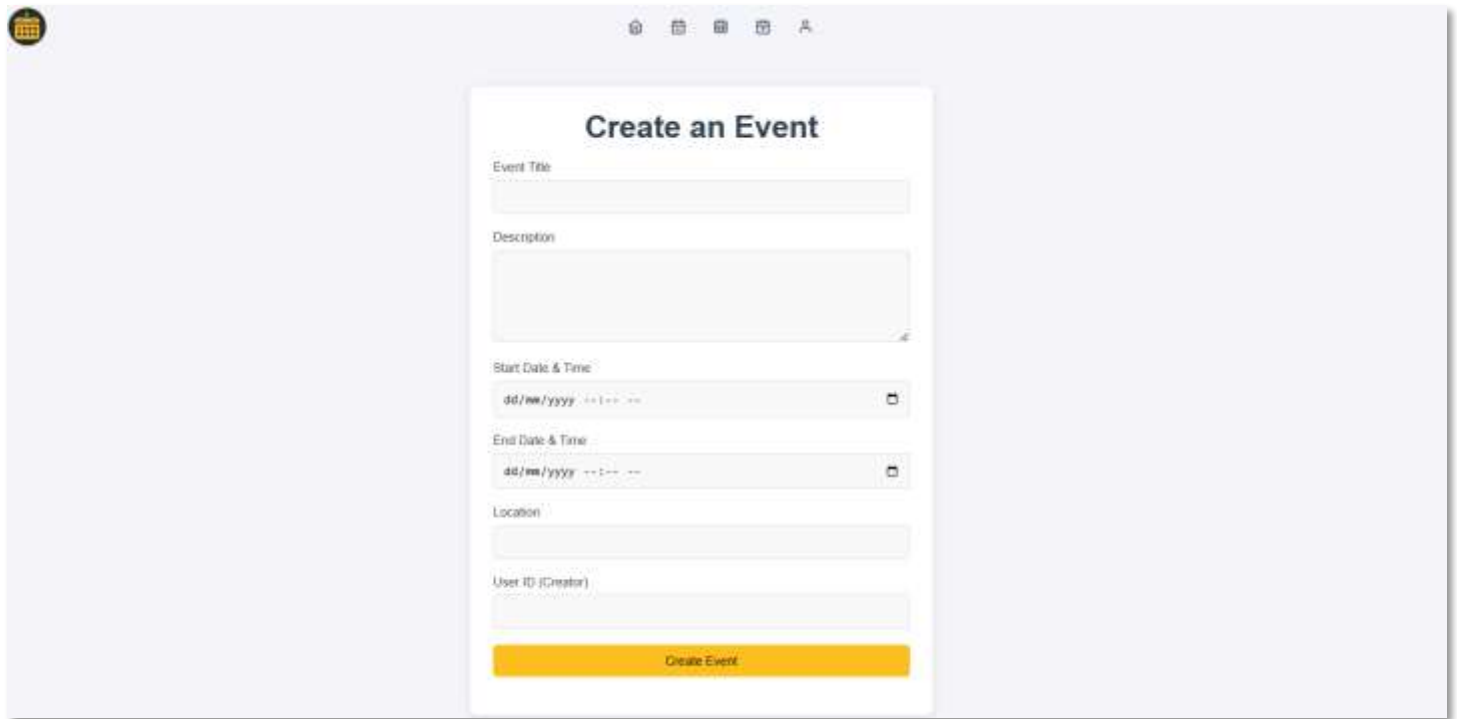
Figure 11: Snippet from the CalendarController.

As shown in figure 11, role-based access controls were implemented in the CalendarController to ensure that users only access webpages unless they are authorised, otherwise they are redirected to prevent unauthorised access. These checks are consistent throughout the controllers to meet the security requirements.

5.3.2 CRUD Frontend

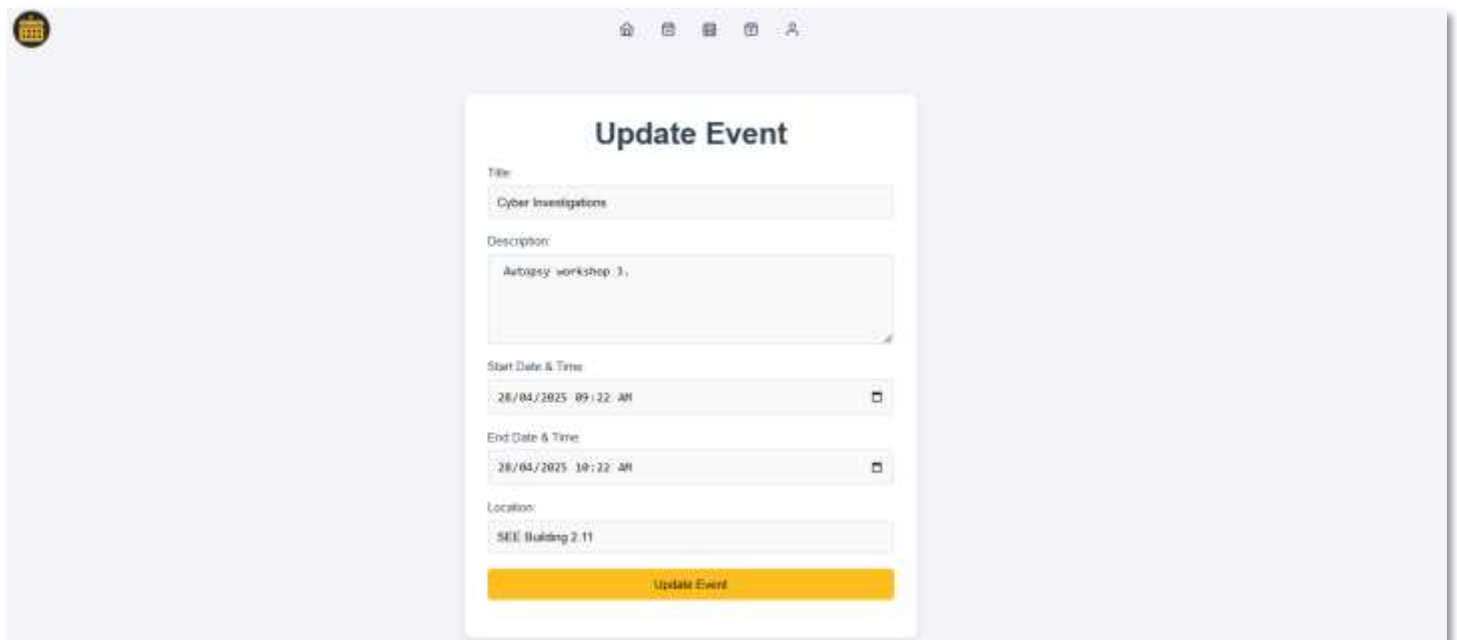
Once the backend was developed, the next requirement that was implemented is the frontend. This displays the event information to the user. Additional functionality is available for admin users; therefore, separate views were created. The user event table displays the event information such as event title, description, start and end times and location. The admin event table includes the additional functionality to delete the event, and update which redirects the admin to the update page. Admins are authorised to access the Create and Update webpages which allows them to create events. Input validation and CSRF tokens have been implemented on the input

fields. The process of implementing and testing input validation against SQL injection is discussed in Section 6.4.4.



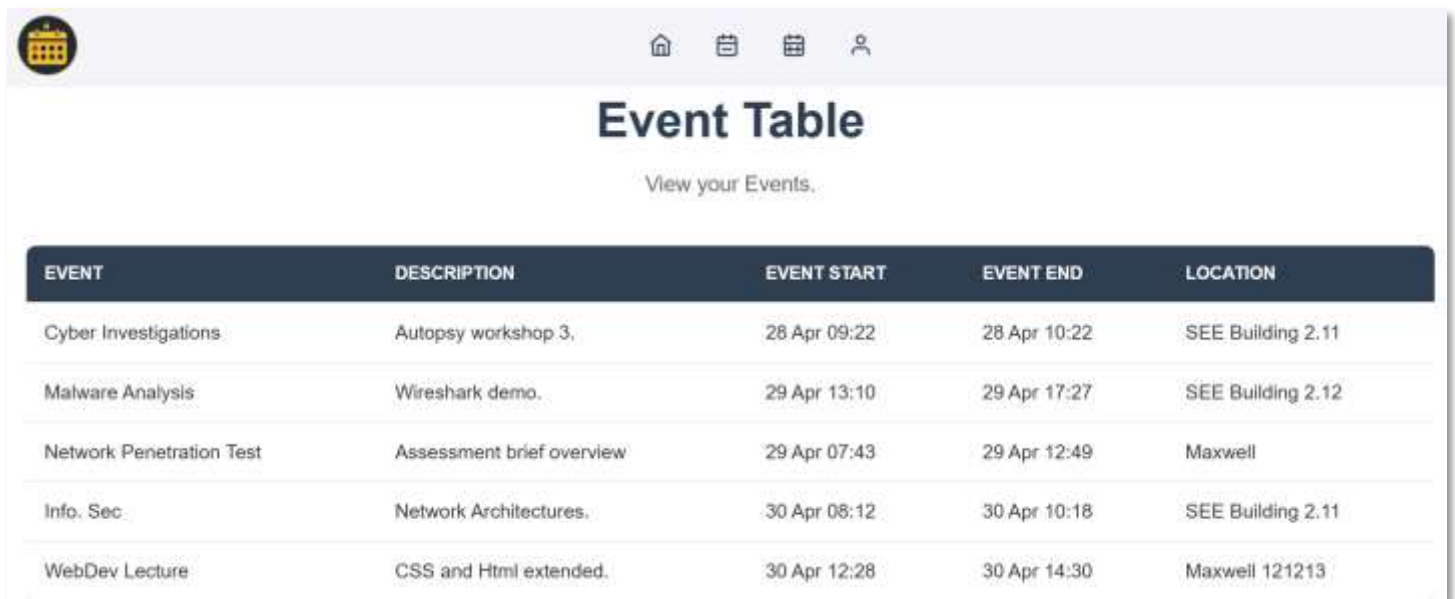
The screenshot shows a web application interface with a top navigation bar containing a calendar icon and four menu icons. The main content area features a central white card titled "Create an Event". The card contains the following form fields: "Event Title" (text input), "Description" (text area), "Start Date & Time" (datetime picker showing "dd/mm/yyyy --:-- --"), "End Date & Time" (datetime picker showing "dd/mm/yyyy --:-- --"), "Location" (text input), and "User ID (Creator)" (text input). A yellow "Create Event" button is at the bottom of the card.

Figure 12: Admins create event page



The screenshot shows the same web application interface, but with a central white card titled "Update Event". The form fields are pre-filled with the following values: "Title" (text input with "Cyber Investigations"), "Description" (text area with "Autopsy workshop 3."), "Start Date & Time" (datetime picker with "28/04/2025 09:22 AM"), "End Date & Time" (datetime picker with "28/04/2025 10:22 AM"), and "Location" (text input with "SHEL Building 2.11"). A yellow "Update Event" button is at the bottom of the card.

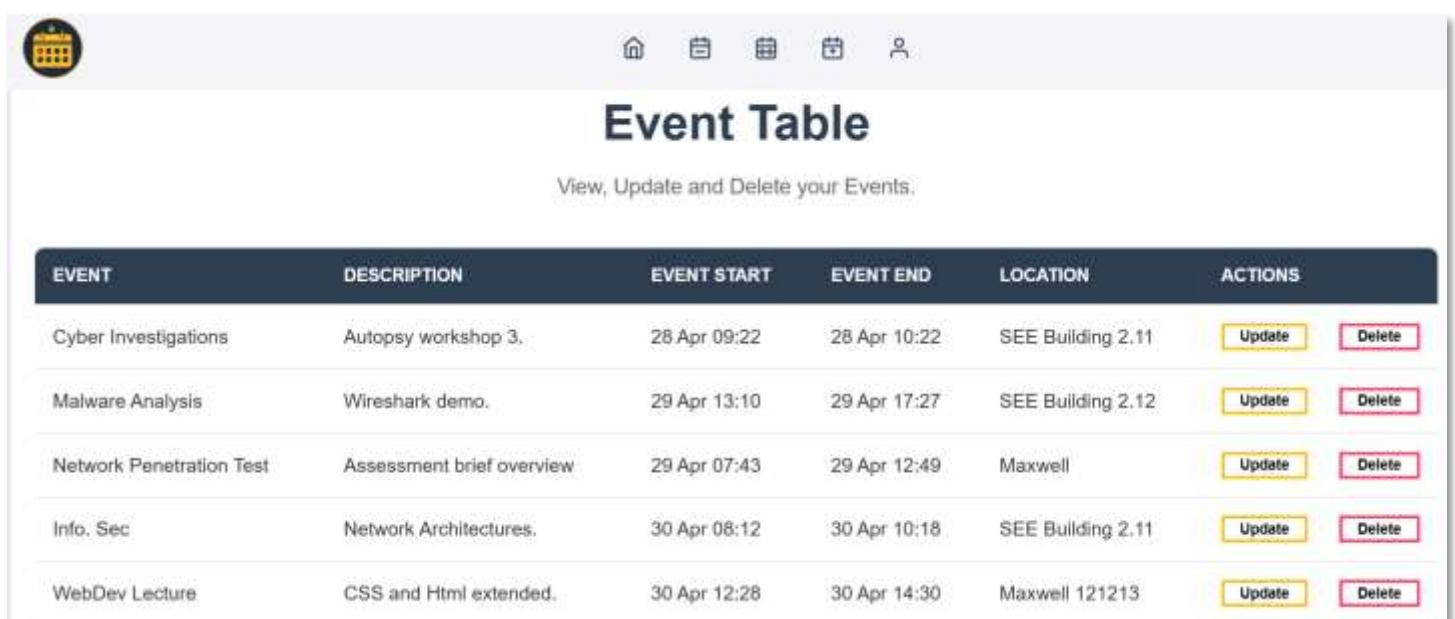
Figure 13: Admins update event page



The screenshot shows a web application interface for a user. At the top, there is a navigation bar with a home icon, a calendar icon, a list icon, and a user profile icon. Below the navigation bar, the title "Event Table" is displayed in a large, bold font. Underneath the title, the text "View your Events." is shown. The main content area contains a table with five columns: EVENT, DESCRIPTION, EVENT START, EVENT END, and LOCATION. The table lists five events: Cyber Investigations, Malware Analysis, Network Penetration Test, Info. Sec, and WebDev Lecture. Each event row contains details about the event's description, start and end times, and location.

EVENT	DESCRIPTION	EVENT START	EVENT END	LOCATION
Cyber Investigations	Autopsy workshop 3.	28 Apr 09:22	28 Apr 10:22	SEE Building 2.11
Malware Analysis	Wireshark demo.	29 Apr 13:10	29 Apr 17:27	SEE Building 2.12
Network Penetration Test	Assessment brief overview	29 Apr 07:43	29 Apr 12:49	Maxwell
Info. Sec	Network Architectures.	30 Apr 08:12	30 Apr 10:18	SEE Building 2.11
WebDev Lecture	CSS and Html extended.	30 Apr 12:28	30 Apr 14:30	Maxwell 121213

Figure 14: User Event table



The screenshot shows a web application interface for an administrator. At the top, there is a navigation bar with a home icon, a calendar icon, a list icon, a calendar icon, and a user profile icon. Below the navigation bar, the title "Event Table" is displayed in a large, bold font. Underneath the title, the text "View, Update and Delete your Events." is shown. The main content area contains a table with six columns: EVENT, DESCRIPTION, EVENT START, EVENT END, LOCATION, and ACTIONS. The table lists five events: Cyber Investigations, Malware Analysis, Network Penetration Test, Info. Sec, and WebDev Lecture. Each event row contains details about the event's description, start and end times, and location, along with "Update" and "Delete" buttons in the ACTIONS column.

EVENT	DESCRIPTION	EVENT START	EVENT END	LOCATION	ACTIONS
Cyber Investigations	Autopsy workshop 3.	28 Apr 09:22	28 Apr 10:22	SEE Building 2.11	Update Delete
Malware Analysis	Wireshark demo.	29 Apr 13:10	29 Apr 17:27	SEE Building 2.12	Update Delete
Network Penetration Test	Assessment brief overview	29 Apr 07:43	29 Apr 12:49	Maxwell	Update Delete
Info. Sec	Network Architectures.	30 Apr 08:12	30 Apr 10:18	SEE Building 2.11	Update Delete
WebDev Lecture	CSS and Html extended.	30 Apr 12:28	30 Apr 14:30	Maxwell 121213	Update Delete

Figure 15: Admin Event table

5.3.3 Problems encountered

Initially the decision to render pages for the different user types caused problems as the events would not display consistently. This was solved by reworking the logic for the user type checks to check the user's role in

the blade templates and routing the logic.

The Edit and Delete buttons would be displayed to users regardless of their role. The access restrictions were enforced with `@if ($user->role == 'admin')` to ensure only admins can perform these actions.

5.4 User authorization and authentication

This requirement included the implementation of the login, register, profile and reset password functionality. CSRF tokens are in each page header to enforce authentication across the webapp.

When first visiting the webapp, users and visitors can only access the homepage which serves as a landing page.

5.4.1 Homepage



Figure 16: Homepage.

Accessing any of the links on the navbar or manually typing page links will reroute the user to the login page to ensure there are no workarounds for unauthorised access. This is enforced via the 'auth' middleware as explained in Section 5.7, which ensures only users that have passed authorisation can access the functionality of the webapp. All webpages were designed using whitespace to improve readability by creating a sense of balance and professionalism in line with the usability and accessibility requirements. (Paralect, 2024)

5.4.2 Login and Register

The UI for the login and register/sign-up pages has been implemented with separate pages. The login page includes links to the registration and password reset pages for better usability. The registration page also includes a link back to the login page, and both forms require a username and password, with the registration page requiring a confirmation of the chosen password. Also included is the 'remember me' checkbox which creates a token that is stored in the database and checked when the same user revisits the webapp, meeting the usability requirements.

Form validation across the webapp is handled by Laravel's server-side authentication system and includes `@csrf` input validation to ensure only the correct formats would be accepted.

```

<x-auth-session-status class="mb-4" :status="session('status')" />

<form method="POST" action="{{ route('login') }}">
  @csrf
  <!-- Email Address -->
  <div>
    <x-input-label for="email" :value="__('Email')"/>
    <x-text-input id="email" class="block mt-1 w-full" type="email" name="email" :value="old('email')" require
    <x-input-error :messages="$errors->get('email')" class="mt-2" />
  </div>

  <!-- Password -->
  <div class="mt-4">
    <x-input-label for="password" :value="__('Password')"/>

```

Figure 17: Snippet from the login backend

Figure 18: Login page

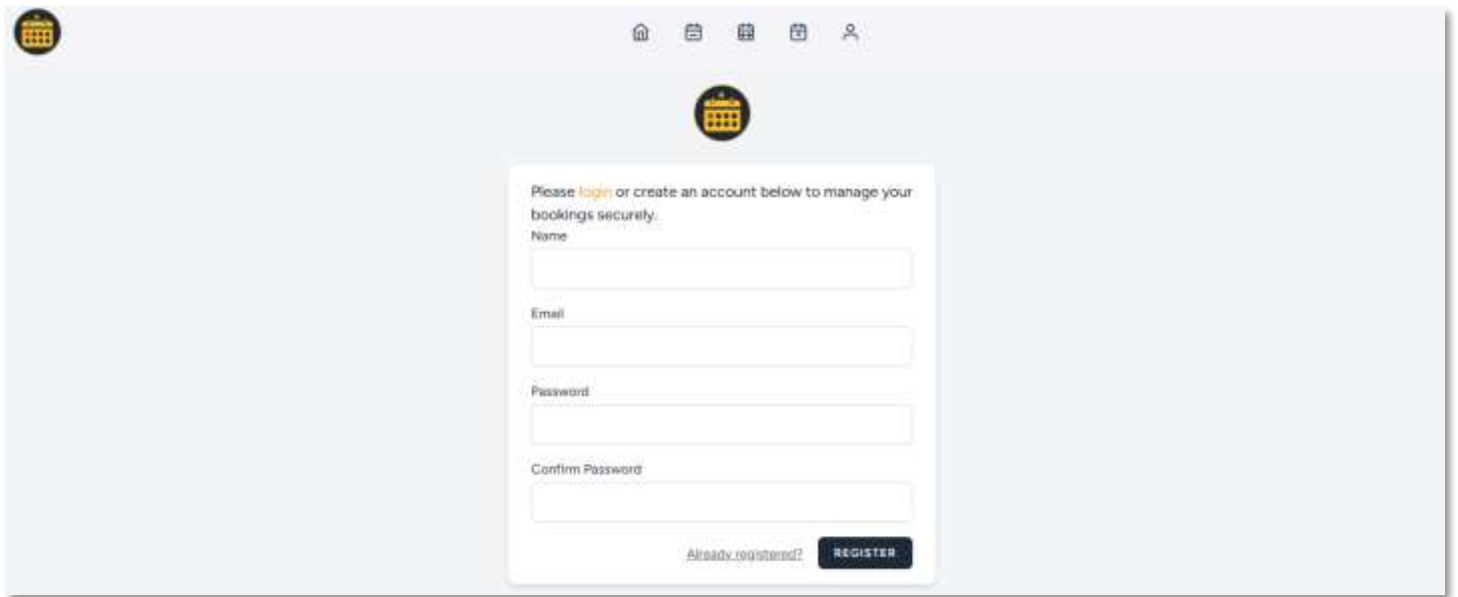


Figure 19: Registration page

5.4.3 Problems encountered

Once displayed on the test server, the blade functions such as `@section`, `@extends` and `@csrf` were displayed as raw text instead of being parsed. This was because the server was not configured properly, and was solved by running Laravel's PHP artisan serve for the web server setup.

Failed login attempts would display the raw error codes from the database which is the main problem this webapp was created to solve. To help prevent username enumeration attacks, the logic was changed to only display generic error messages to the user for failed logins.

5.5 Account management

Account management functions have been implemented on a single Profile page. It includes profile information for users to review and update their account details such as their name, email address and password. The logic for all the forms is within routes, such as `password.update: Route::put('password', [PasswordController::class, 'update'])->name('password.update');`.

```

class ProfileController extends Controller
{
    public function update(ProfileUpdateRequest $request): RedirectResponse
    {
    }

    /**
     * Delete the user's account.
     */
    /**usage
    public function destroy(Request $request): RedirectResponse
    {
        $request->validateWithBag('userDeletion', [
            'password' => ['required', 'current_password'],
        ]);

        $user = $request->user();

        Auth::logout();

        $user->delete();

        $request->session()->invalidate();
        $request->session()->regenerateToken();

        return Redirect::to('path: '/'');
    }
}

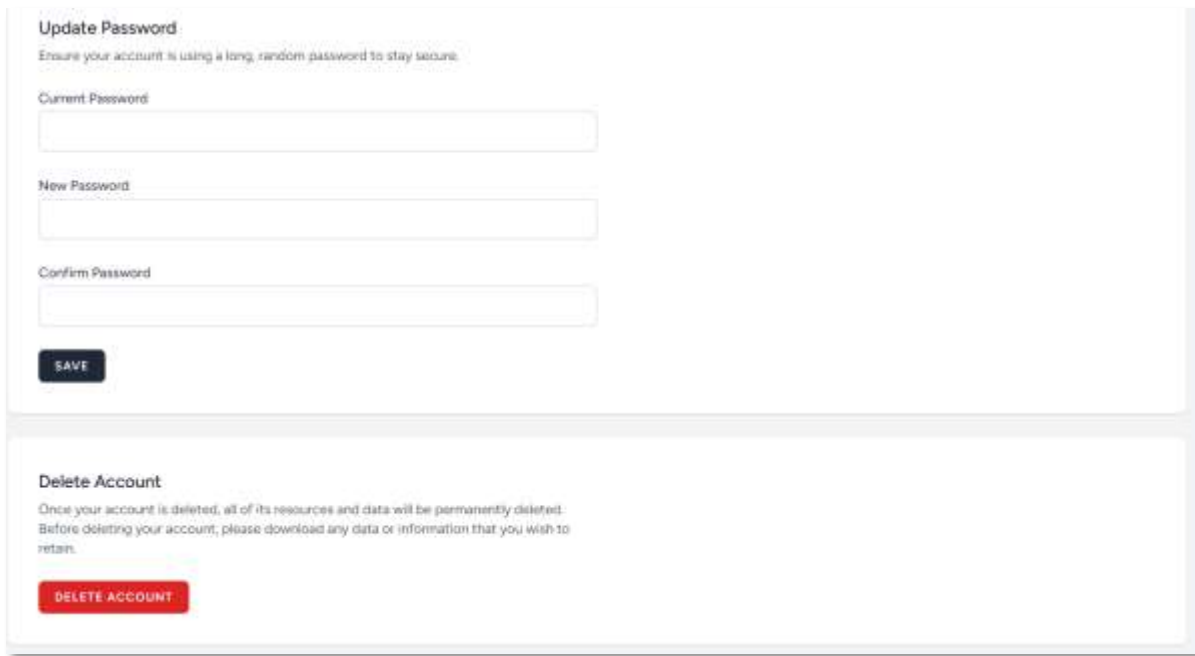
```

Figure 20: Snippet from ProfileController

Along with the functions to view and update details, users can also delete their accounts. This controller is part of Laravel's authorisation. It invalidates the user's session and regenerates the session token before redirecting the user to the homepage.

The screenshot shows a web application interface for a user profile. At the top, there's a header with a logo on the left and the user's name 'Huzaiifa Patel' on the right. Below the header, there's a 'Profile' section with a 'Log Out' button. The main content area is divided into two sections: 'Profile Information' and 'Update Password'. The 'Profile Information' section has a subtitle 'Update your account's profile information and email address.' and two input fields: 'Name' (containing 'Huzaiifa Patel') and 'Email' (containing 'huzaiifapatel2003@outlook.com'). Below these fields is a 'SAVE' button. The 'Update Password' section has a subtitle 'Ensure your account is using a long, random password to stay secure.' and a label for 'Current Password'.

Figure 21: Profile page



The screenshot shows a web interface for a profile page. It is divided into two main sections. The top section is titled "Update Password" and includes a sub-header "Ensure your account is using a long, random password to stay secure." Below this are three input fields: "Current Password", "New Password", and "Confirm Password". A dark "SAVE" button is positioned at the bottom of this section. The bottom section is titled "Delete Account" and includes a sub-header "Once your account is deleted, all of its resources and data will be permanently deleted. Before deleting your account, please download any data or information that you wish to retain." Below this is a red "DELETE ACCOUNT" button.

Figure 22: Profile page continued

```
Route::get( uri: '/dashboard', function () {  
    return view( view: 'dashboard');  
})->middleware(['auth', 'verified'])->name( name: 'dashboard');
```

Figure 23: Snippet from web.php.

As mentioned in Section 5.7, middlewares are in place at all requests to ensure only authorised users can access the Profile/dashboard page to update their account details.

5.5.1 Password Reset

Password reset functionality was created next. This page firstly checked the user's session status, created a csrf token then presented the form. The form has input validation and routes to 'password.email'.


```

<div class="mb-4 text-sm text-gray-600">
    {{ __('Forgot your password? No problem. Enter your email address below and we will email you a password reset link.') }}
</div>

<!-- Session Status -->
<x-auth-session-status class="mb-4" :status="session('status')" />

<form method="POST" action="{{ route('password.email') }}">
    @csrf

    <!-- Email Address -->
    <div>
        <x-input-label for="email" :value="__('Email')" />
        <x-text-input id="email" class="block w-full" type="email" name="email" :value="old('email')" required />
        <x-input-error :messages="errors->get('email')" class="mt-2" />
    </div>

    <div class="flex items-center justify-end mt-4">
        <x-primary-button>
            {{ __('Email Password Reset Link') }}
        </x-primary-button>
    </div>
</form>

```

Figure 24: Password reset page backend

The environment variable (.env file) was updated with an email address and the Gmail app password which allowed the Laravel's built-in password reset system to use this email address to send out password reset emails. This gives users a secure and user-friendly way to regain access to their accounts, improving the usability of the web app.

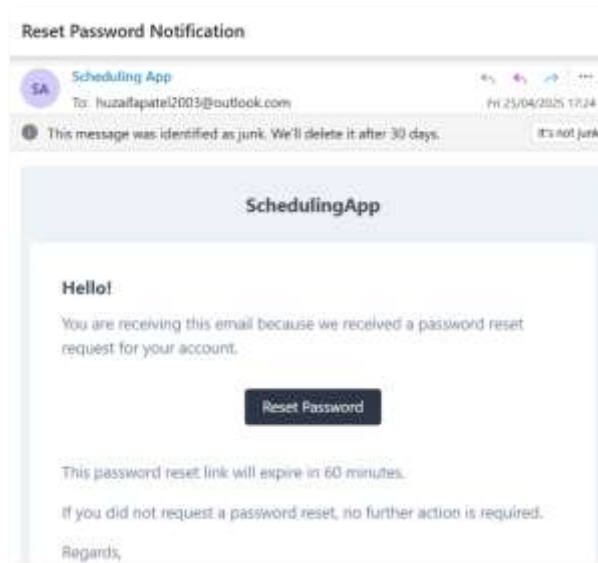


Figure 25: Password Reset Email.

This email includes a unique token with a time limit, and a secure link to the web app. The token is hashed for security and stored in the password_reset_tokens table in the database as shown below and includes the creation time.

email	token	created_at
@outlook.com	\$2y...	2025-04-25 16:24:27
test@test.com	\$2y:/tTzw0...	2025-02-04 16:20:22

Figure 26: Database table password_reset_tokens

5.5.2 Problems encountered

The token expiration time was initially too short which was received in the feedback during user testing. This resulted in the tokens expiring before user had the chance to use them, locking the user out of their account.

The UI was broken for the password reset page because of this the form submission was not working. This was solved with step-by-step debugging which revealed that the Blade template for the webpage was incorrectly referenced.

5.6 Calendar Displays

To meet the project requirements, additional views were created to display the events in an interactive calendar format. The open source 'FullCalendar' was used for this purpose, implemented via JavaScript. The display is currently set to show events in the current week and users can change the weekly display using the arrows for better usability.

Events from the database were fetched via a controller method then passed onto the view where the events were displayed by FullCalendar using the JSON format.

```

<div class="calendar-container">
<div class="calendar-full">
<div id="calendar" class="calendarfull">
<script src="https://cdn.jsdelivr.net/npm/fullcalendar@6.1.0/index.global.min.js"></script>
</script>

document.addEventListener('DOMContentLoaded', function () {
  var calendarEl = document.getElementById('calendar');
  var calendar = new FullCalendar.Calendar(calendarEl, {
    initialView: 'timeGridWeek', //changes the view to day, weekly or monthly view
    events: {!! json_encode($events) !!}, // Gets the events from calendar controller
    editable: true, // Allows drag-and-drop
    droppable: true,
    drop: function(info) {
      saveNewEvent(info);
    },
    eventClick: function(info) {
      let event = info.event;

      // Display the event's details when clicked
      alert('Event: ' + event.title + '\nStart: ' + event.start.toLocalString() + '\nEnd: ' + event.end.toLocalString() + '\nDescription: ' + event.description);

      if (confirm('Do you want to edit this event?')) {
        window.location.href = info.event.url;
      }
    }
  });
  calendar.render();
});

```

Figure 27: Calendar backend

5.6.1 User Calendar

As shown in the calendar backend, users can click on events which will open a dialogue box and display the event information. Admins have the added functionality of editing the event via a link from this dialogue box, whereas users are only able to view their events in this display.

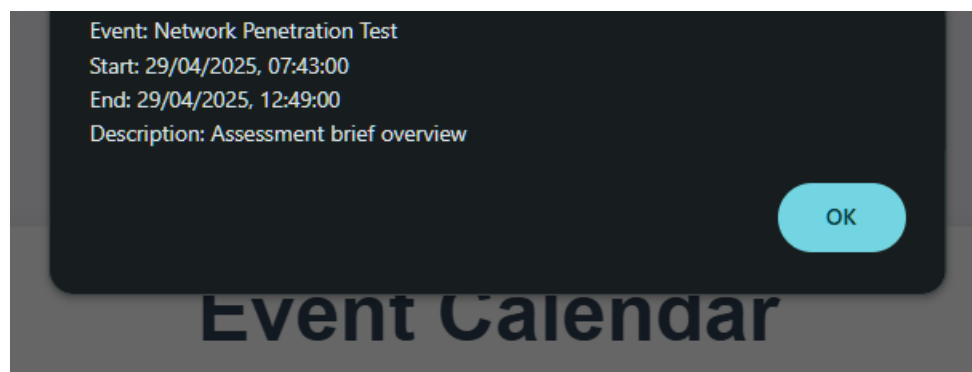


Figure 28: Calendar event information

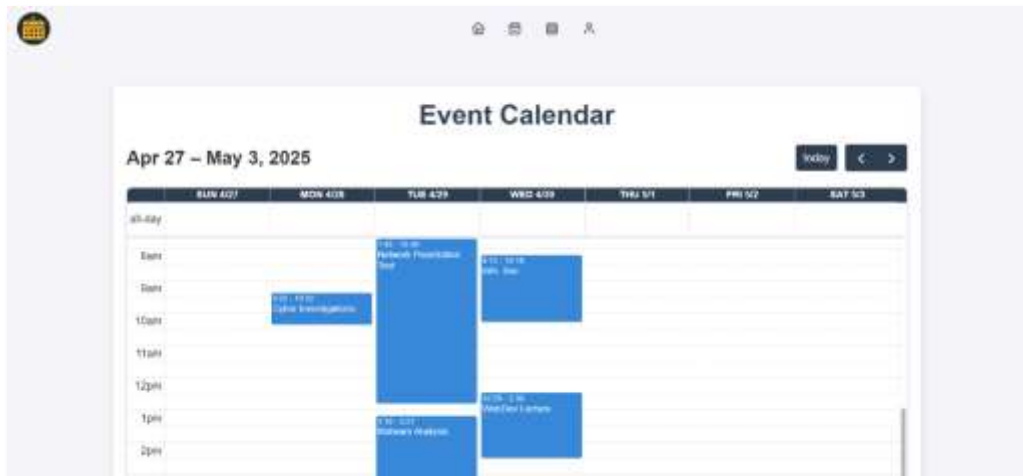


Figure 29: User Calendar view.

5.6.2 Admin Calendar

Admins have the additional functionality of updating events. An interactive sidebar was implemented to allow admins to drag-and-drop pre-determined events for better usability, as well as a shortcut link to create a new event. These calendar views enhance usability and improve accessibility for different types of users.

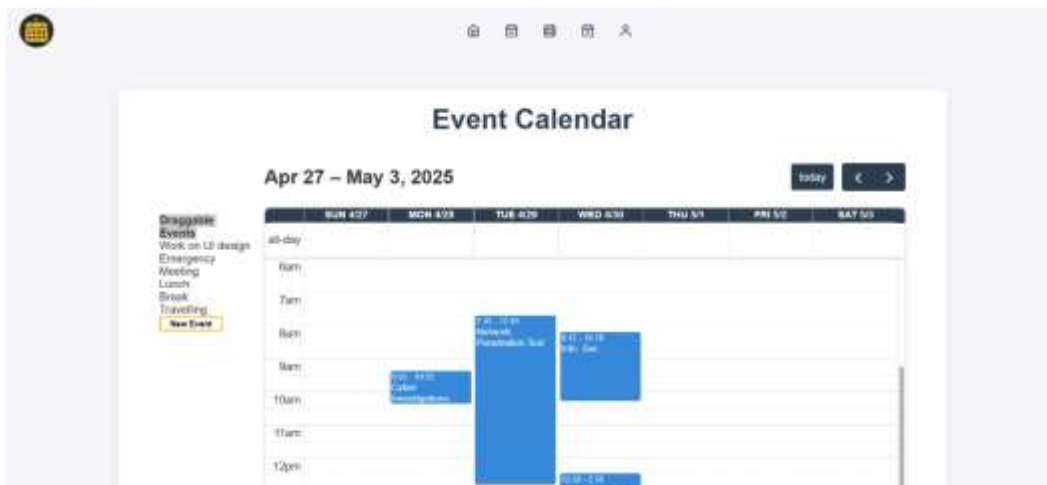


Figure 30: Admin Calendar view.

5.6.3 Problems encountered

FullCalendar was initially implemented via CDN in the blade templates which caused performance issues. Instead, the FullCalendar script was input directly to the webpage which solved this issue.

The interactive sidebar for draggable events would show the new events on the calendar, however, it would not result in any changes to the database. This was solved by ensuring the AJAX request was sent to the backend where the /calendar/store route accepted the POST request once the event was dropped. This change would now go through to the backend and result in changes to the database as intended.

5.7 Custom error handling

As discussed in Section 5.4.1, Accessing any of the links on the navbar or manually typing page links will reroute the user to the login page to ensure there are no workarounds for unauthorised access. Additional error handling was implemented to ensure users would not receive detailed error messages when they manually searched for a non-existent webpage. A new route was introduced to force users to the custom error page when this happens.

```
// Returns a standard error page.  
Route::fallback(function () {  
    return view('433');  
});  
require __DIR__.'/auth.php';
```

Figure 31: Fallback function route.

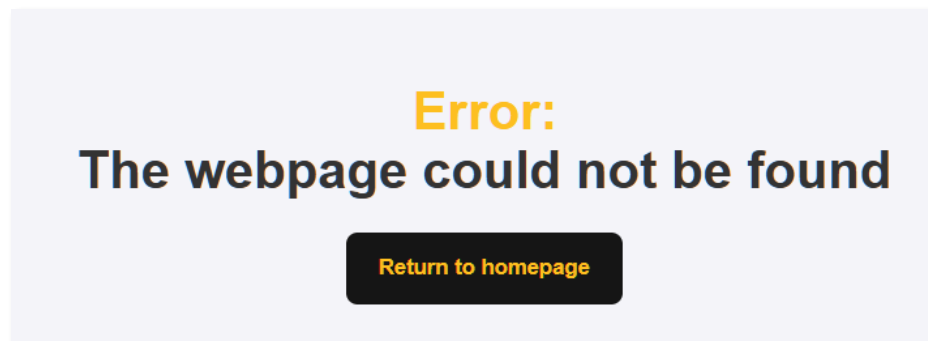


Figure 32: Custom error handling- 404 errors.

5.8 Middleware

Middleware in Laravel filters the HTTP requests sent by the user. In this project these are the requests sent to fulfil the CRUD operations for the events. Middleware was used to apply the logic before and after the requests were handled by the controller. The main benefit of using middleware is it allowed me to enforce security and access controls to ensure that only authenticated users can access certain routes.

```
Route::middleware('auth')->group(function () {
    //allows routing calendar page from nav bar and the url: /calendar
    Route::get('url: '/calendar', [CalendarController::class, 'index'])->name('calendar.index');
    Route::get('url: '/calendar/user', [CalendarController::class, 'userindex'])->name('calendar.userindex');
    Route::get('url: '/calendar/admincalendar', [CalendarController::class, 'calendar'])->name('calendar.admin');
    Route::get('url: '/calendar/calendar', [CalendarController::class, 'calendar'])->name('calendar.calendar');
```

Figure 33: Snippet from web.php.

All requests went through 'auth' authentication middleware to ensure that only logged-in users can view events, and that only admin users can perform CRUD operations such as creating, viewing, editing, or deleting events.

Role-based middleware restricted access to unauthorised users.

```
Route::middleware('auth')->group(function () {
    //allows routing calendar page from nav bar and the url: /calendar
    Route::get('url: '/calendar', [CalendarController::class, 'index'])->name('calendar.index');
    Route::get('url: '/calendar/user', [CalendarController::class, 'userindex'])->name('calendar.userindex');
    Route::get('url: '/calendar/admincalendar', [CalendarController::class, 'calendar'])->name('calendar.admin');
    Route::get('url: '/calendar/calendar', [CalendarController::class, 'calendar'])->name('calendar.calendar');

    //creating and deleting events
    Route::get('url: '/calendar/create', [CalendarController::class, 'create'])->name('calendar.create');
    Route::post('url: '/calendar/store', [CalendarController::class, 'store'])->name('calendar.store');

    Route::delete('url: '/calendar/{id}', [CalendarController::class, 'destroy'])->name('calendar.destroy');
    //editing and updating events
    Route::get('url: '/calendar/{id}/edit', [CalendarController::class, 'edit'])->name('calendar.edit');
    Route::put('url: '/calendar/{id}', [CalendarController::class, 'update'])->name('calendar.update');
    //user info updates
```

Figure 34: Authorisation middleware

The authorisation middleware was used on all the CRUD operations and the other functions available to admin users as an extra layer of security, ensuring unauthorised access does not occur.

5.8.1 Problems encountered

Once implemented, the authorisation middleware caused issues that rendered the web app unusable as users could not access or edit any information from the database. This was because multiple middleware and routes were checking authorisation at once and was solved by reviewing the logic and rebuilding the routes by following the Laravel documentation. (Laravel 2025)

5.9 Summary

In this section the development and implementation of the requirements and designs from Section 4 were discussed. The requirements and designs were successfully implemented in this project and a sufficient minimum viable product was achieved.

Chapter 6: Testing and Analysis

6.1 Introduction

Following the development and implementation of the requirements, this chapter provides an overview of the web application testing process. In this phase the developed product is tested on whether it meets user requirements. (Shylesh, 2017) Testing has been performed to guarantee the code is efficient and reliable. Various methods of testing were performed for both usability and security.

These following methods were used: regression testing, penetration testing, web testing and usability testing.

The testing strategy was to not only test the web application, but also to review the results and consistently work on improvements in the following sprints. Section 6.4.5 provides an example of critical problems that were found during the penetration testing process, along with the actions taken because of the testing process.

6.2 Regression testing

Regression testing guarantees that all changes to the software, UI or backend functionality or removing functionality does not impact the existing functionality. Regression testing also tests whether the software or the application works properly after the fixation of any bugs or errors. (Jamil et al., 2016) Regression testing was performed manually after each new functionality had been implemented. In this project regression testing was used to ensure that the code that was already written was not negatively affected by the new changes to the code once they had been introduced.

As discussed in Section 5.4.2.1, once the login pages were implemented and run on the test server, the blade functions such as `@section`, `@extends` and `@csrf` were displayed as raw text instead of being parsed.

```
@section('title', 'Login')
@csrf
☐ {{ __('Remember me') }}
{{ \_\_\('Register'\) }} @if (Route::has('password.request')) {{ \_\_\('Forgot your password?'\) }} @endif {{ __('Log in') }}
```

Figure 35: Raw code displayed on the web app

This was solved by making the necessary changes to the code. Once the changes were made, regression testing was performed on the new login page code to ensure the Blade view is being compiled correctly as shown below.


```

1 <?php
2 test(description: 'login page renders correctly', function () {
3     $response = $this->get('/login');
4
5     $response->assertStatus(200);
6
7     // Check raw Blade syntax is not visible
8     $response->assertDontSee('@extends');
9     $response->assertDontSee('@section');
10    $response->assertDontSee('{{');
11    $response->assertDontSee('}}');
12
13    // Check that expected text IS visible
14    $response->assertSee('Log in');
15    $response->assertSee('Email');
16    $response->assertSee('Password');
17    $response->assertSee('Register');
18
19 });

```

Figure 36: Regression testing the login page

This test makes a GET request to /login, ensures the page loads correctly then confirms the Blade directives (@extends, @section and {{}} do not show. It also confirms the body text and hyperlinks for ‘Log in’ and ‘Register’ are visible. This is an example of regression testing that was performed using a test case, however a great majority of the issues related to new code implementation were solved with manual regression testing. The process followed for manual regression testing is displayed in figure 37.

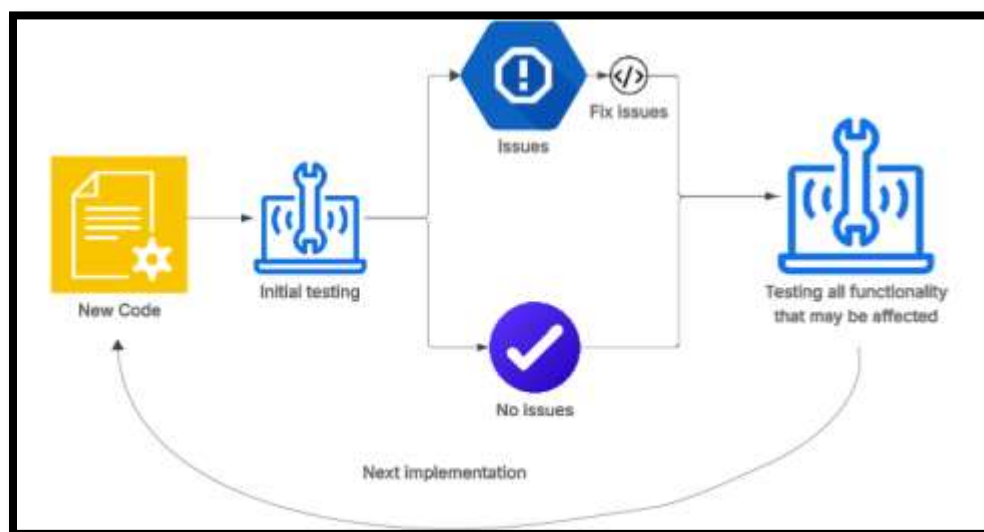


Figure 37: Regression testing process

6.3 Usability testing

Usability testing is the process of software, or a service being tested by its target user demographic. The purpose of usability testing is to determine whether the project meets the initially specified requirements of the user or not. (Jamil et al., 2016) Once testing has been carried out, the results are reviewed to determine the areas of improvement to enhance the project. The ease or difficulty that users experience with systems will determine their success or failure. (Insfran, Fernandez, 2008)

The usability testing was conducted by 8 volunteers, who were requested to use the website as they would use any other website. They were then asked to provide feedback on their overall experience. The questions asked were based on the project's requirements in section 4.

The following questions were asked:

Question 1: Overall, how easy or difficult was it to navigate the website? Score 0 (Difficult) - 5 (Easy)

Question 2: Was the website design similar to those you have used before? Yes or No.

Question 3: Is the Events Table page easy to understand and navigate? Yes, or Not easy to understand or Not easy to navigate.

Question 4: Is the Calendar page easy to understand and navigate? Yes or Not easy to understand or Not easy to navigate.

Question 5: How was the overall user experience on the website? Score 0 (Negative) - 5 (Positive)

Question 6: How was the performance of the website? Score 0 (Negative) - 5 (Positive)

6.4 Penetration testing

Research was conducted to find suitable tools to carry out this penetration test. The OWASP list of vulnerability scanning tools for Dynamic Application Security Testing (DAST) was the main source for these tools, in line with the project's objective to defend against the OWASP top threats. (OWASP, 2024) Burp Suite was the chosen tool for penetration testing. I have leveraged both manual and automated tools to find vulnerabilities as I had planned to in Section 3 – Methodology.

6.4.1 Virtual Machine Setup

At this stage, the project had been hosted locally and only accessible to the host machine (my computer) by visiting the URLs: localhost:8000 or 127.0.0.1:8000. This is the industry standard for locally testing projects; however, I would need to access the webapp using a different machine to test the safeguards in place.

Many considerations were made about how to host the project during this phase. On Kali Linux, Inetsim was used to simulate internet services. Inetsim is a powerful tool that allows internet services to be simulated, acting as a fake or dummy server to capture network activity. As discussed in Section 4.3.2.1, when hosted on

XAMPP, the web application is limited to localhost and not exposed to the internet which makes it suitable for hosting during security testing. This is important as only this web application will be affected if the server crashes or becomes corrupted during security testing. Penetration testing a live web application can result in crashing the hosting platform e.g. GoDaddy and affecting the other services hosted online which has both ethical concerns and legal consequences. (SoftwareSecured, n.d.)

The virtual machine was configured on the same network as my host computer, which had network segmentation in place to ensure a safe and secure environment for penetration testing. Testing my own project in this controlled test environment is the safest method.

```
tSim/DNS.pm line 69.  
* finger_79_tcp - started (PID 26881)  
* daytime_13_udp - started (PID 26887)  
* tftp_69_udp - started (PID 26878)  
* ident_113_tcp - started (PID 26882)  
* chargen_19_tcp - started (PID 26894)  
* time_37_tcp - started (PID 26884)  
* ftp_21_tcp - started (PID 26876)  
* http_80_tcp - started (PID 26870)  
* echo_7_udp - started (PID 26889)  
* quotd_17_udp - started (PID 26893)  
* discard_9_tcp - started (PID 26890)  
* time_37_udp - started (PID 26885)  
* ftps_990_tcp - started (PID 26877)  
* quotd_17_tcp - started (PID 26892)  
* https_443_tcp - started (PID 26871)  
* chargen_19_udp - started (PID 26895)  
* pop3s_995_tcp - started (PID 26875)  
* ntp_123_udp - started (PID 26880)  
* dummy_1_tcp - started (PID 26896)  
* daytime_13_tcp - started (PID 26886)  
* echo_7_tcp - started (PID 26888)  
* dummy_1_udp - started (PID 26897)  
* syslog_514_udp - started (PID 26883)  
done.  
Simulation running.
```

Figure 44: Inetsim's dummy server simulation.

6.4.2 Brute force attack

Firstly, the localhost webapp was successfully accessed on the virtual machine.

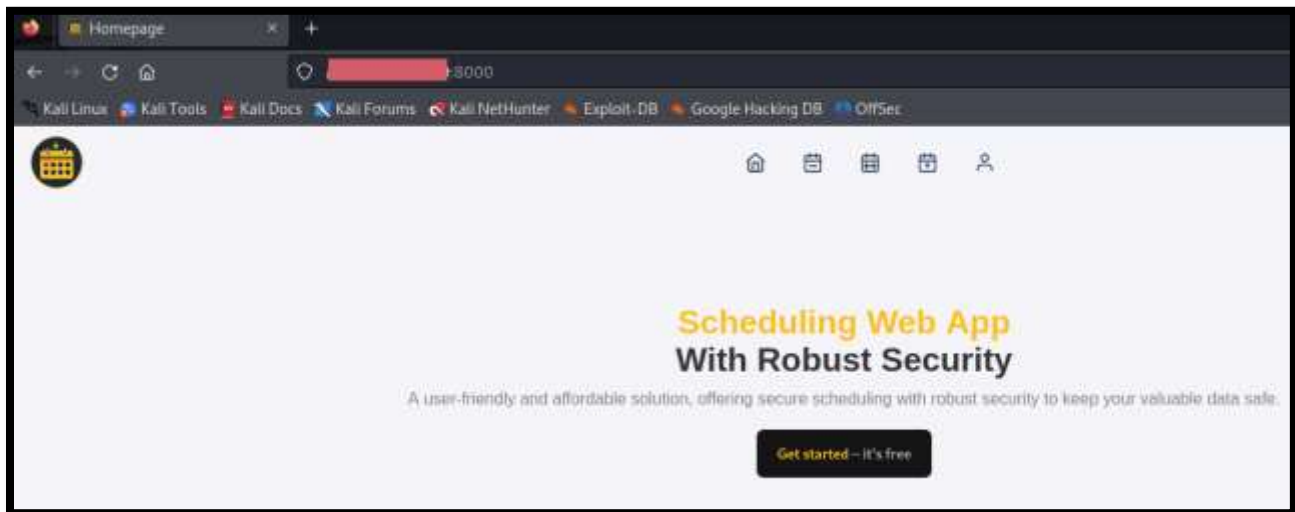


Figure 45: Accessing the webapp using a Virtual Machine.

Burp Suite was used to target the webapp with a brute force attack using a generic list of passwords. Firstly, the login POST request was intercepted in Burp Suite. Next, a payload of generic passwords was loaded and executed as a 'sniper' attack in the Intruder window. As shown in figure 46, the server accepted the first 5 passwords, then stopped responding to requests.

Intruder attack results filter: Showing all items

Request ^	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		302	1295			1570	
1	123456	302	1234			1570	
2	password	302	1214			1570	
3	12345678	302	1162			1570	
4	qwerty	302	1164			1570	
5	123456789		0				

Figure 46: Testing against brute force attacks

```
Route::post('uri: '/login', [AuthenticatedSessionController::class, 'store'])
->middleware(['guest', 'throttle:5,1'])
->name('name: 'login');
```

Figure 47: Login route

The login POST route is wrapped inside throttle middleware which gives user a maximum of 5 login attempts before they are automatically locked out of the login screen for 1 minute. An error message is then displayed to the user warning 'too many requests' have been sent.

6.4.3 CSRF Vulnerability

In the profile section, users can change their email address. This was the target vector for testing against CSRF vulnerabilities. This requires the attacker to know all the parameters that will go through once the email change

is requested. However, as shown in figure 48, a CSRF token was generated which ensures only authorised requests can pass through. The attack was unsuccessful as the input fields for editing or changing information are all protected with CSRF tokens.

```
_token=HPuqiH0L3sYE60U06ckwLztVW9RNa9IYl3W4xf0J&_method=patch&name=Test+User&email=test%40test.com
```

Figure 48: Email change request

6.4.4 SQL Injection

Input validation is present on all input fields in the webapp. Figure 49 shows one of the SQL Injection attacks. These attacks were unsuccessful as the input field will not accept anything other than a valid email address.



Figure 49: SQL injection login attempt

Attempts to perform SQL injection included using 'test@example.com' OR 'a'='a', 'test@example.com' OR 1=1—' and many other payloads. The SQL injection cheat sheet from 'invicti' was used for testing purposes, which includes technical information and payloads for SQL injection attacks against MySQL databases. (Mavituna, 2022) URL encoding was also attempted, which encodes the apostrophe in URL encoding format %27. The payload 'test%40example.com%27%20OR%201%3D1%20—' was also rejected by the input validation.

6.4.5 SQL Injection – Problems encountered

Testing the 'create event' functions revealed that these inputs were not sanitised as well as the login fields. The following changes were made to increase the security of the web application.

Forms were validated to only accept the expected inputs. Input fields such as Title and Location now accept only letters, full stops, numbers and spaces. This client-side validation helps to prevent SQL Injection attacks whilst allowing the website to function as intended for authorised users.

```
<div class="form-group">
  <label for="location">Location</label>
  <input type="text" id="location" maxlength="25" name="location" class="form-input" required pattern="^[a-zA-Z0-9\.\s]+$">
</div>
```

Figure 50: Form Validation parameters.

Create an Event

- The description field must only contain letters and numbers.

Figure 51: Form Validation error message.

The CalendarController was also updated to include error messages as shown in figure 51 and ensure only alphanumeric characters are submitted. This caused problems as spaces are excluded from the 'alpha_num' rule in Laravel. The workaround was to implement the rule (/^[a-zA-Z0-9\s]+\$/) to exclude all characters except alphanumeric and spaces in CalendarController as shown in figure 52.

```
public function store(Request $request)
{
    // Input validation
    $validatedData = $request->validate([
        'title' => 'required|string|max:30|regex:/^[a-zA-Z0-9\s]+$/ ',
        'description' => 'nullable|string|max:200|regex:/^[a-zA-Z0-9\s]+$/ ',
        'start_datetime' => 'required|date',
        'end_datetime' => 'required|date|after:start_datetime',
        'location' => 'nullable|string|max:30|regex:/^[a-zA-Z0-9\s]+$/ ',
        'user_id' => 'required|integer|exists:users,id', // checking if the id exists in users
    ]);

    // Create the calendar event
    $event = CalendarModel::create([
        'user_id' => $validatedData['user_id'],
        'title' => $validatedData['title'],
        'description' => $validatedData['description'],
        'start_datetime' => $validatedData['start_datetime'],
        'end_datetime' => $validatedData['end_datetime'],
        'location' => $validatedData['location'],
    ]);

    return redirect()->route('calendar.index'); // redirect to calendar page after storing
}
```

Figure 52: Backend validation

These improvements will now enforce both client-side and backend validation before the calendar event is created. The web app was penetration tested once again, resulting in further SQL injection attacks against the create event function proving unsuccessful.

6.5 Web testing

6.5.1 Introduction

This section outlines the results from the markup and CSS validation, as well as independent testing to ensure the website functionality performs across multiple browsers and platforms.

6.5.2 Markup and CSS validation

Validation was performed using <https://validator.w3.org/>. The purpose of validation is to catch unintended mistakes that I might have otherwise missed so that I can fix the potential problems for accessibility, usability, interoperability, security, or maintainability. (validator.w3.org, n.d.)



Figure 53: CSS validation

Validating the CSS file returned no errors, except the false positives shown in figure 53. The `@tailwind base;`, `@tailwind components;`, `@tailwind utilities;` directives are not standard CSS rules. Therefore, these directives have been mis-flagged as errors by the checker.

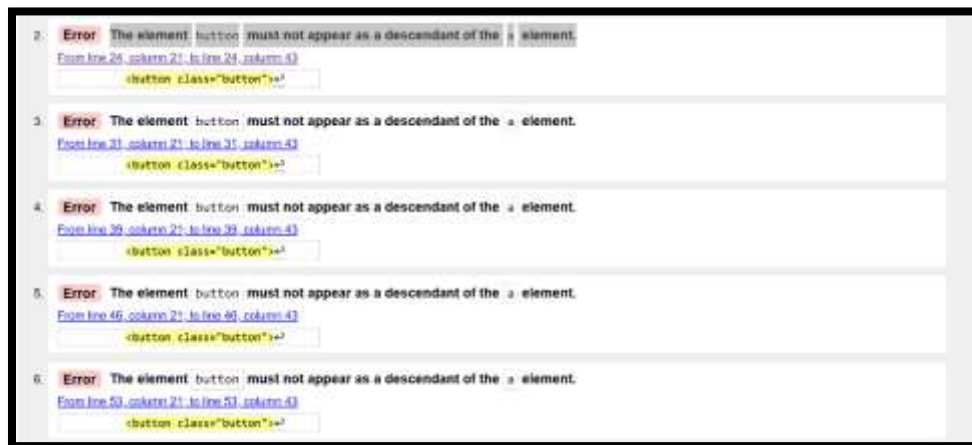


Figure 54: HTML validation errors

Validating the webpages resulted in errors for a missing `<!DOCTYPE html>` tag and incorrect syntax for the navigation bar. This was solved by rewriting the code for the navbar for the guest layout, user layout and admin layout, and removing the unnecessary button elements.

6.5.3 Independent browser and platform functional testing

This section demonstrates the results from testing the web application's functionality on Windows, Linux and IOS platforms. On these platforms, the web application was tested using the Windows Edge, Google Chrome,

Safari and Mozilla Firefox browsers to ensure compatibility and promote accessibility for all types of users. This testing brought awareness to problems that had not been noticed on the Windows platform that was used for development. This included broken links and malfunctional UI, and these issues were solved by following the Laravel documentation and using these best practices in my code.

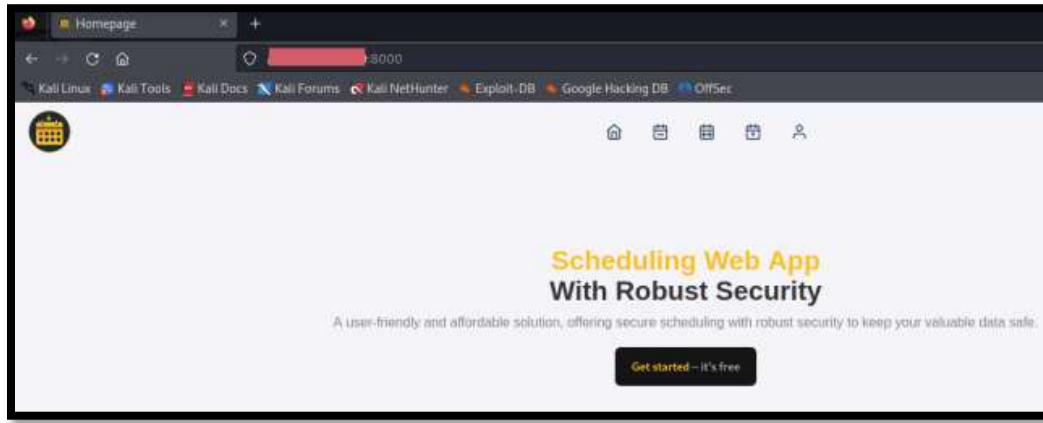


Figure 55: Cross browser testing – Mozilla Firefox using a Linux system.

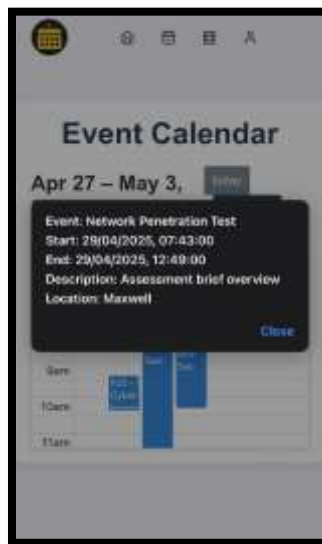


Figure 56: Mobile testing- Safari browser using an IOS system.

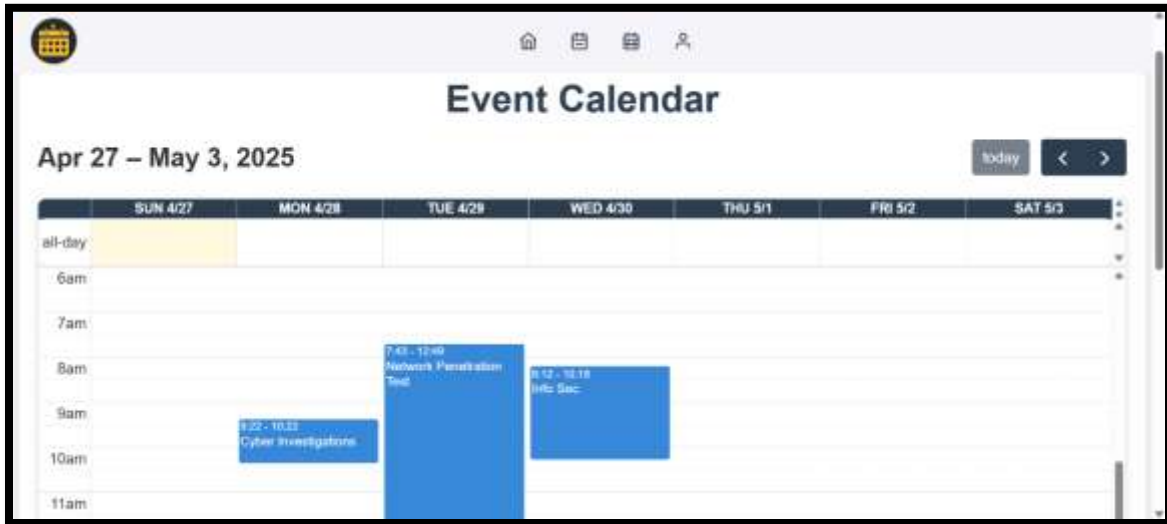


Figure 57: Cross browser testing – Microsoft Edge using a Windows system.

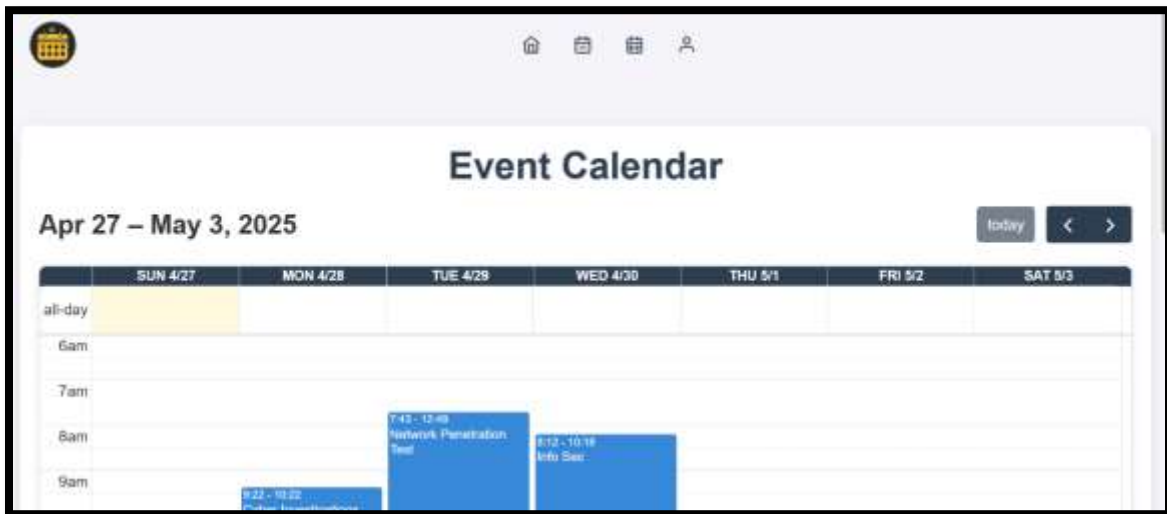


Figure 58: Cross browser testing – Google Chrome using a Windows system.

6.6 Analysis of the testing results

Overall, the results from question 1 show that the website was very easy to navigate, however, 2 volunteers thought there was room for improvement. The website design was adapted from the design phase to resemble other scheduling websites, and this is reflected by the responses to question 2 where the volunteers found the web application similar to scheduling services they have used previously.

The questions related to the ease of navigation on the events table and calendar pages suggest that overall, the calendar page is the preferred view, however it was decided that both pages would be available on the website to cater for a wider audience. The user experience for most users was very positive with one user giving the website a 3 out of 5. The responses from users suggest that the performance of the website is also good.

Overall, the responses to the questions show that users had a very positive experience with most users finding it easy to understand and navigate. The feedback from this questionnaire were addressed. The calendar view was the preferred view for most users. In response to this, the webpage has been updated to include the options to view or edit events and quick-add events via the sidebar to enhance the user experience and improve the functionality available to admin users.

The web and penetration testing results show that the project has met the objectives regarding web security and accessibility.

Chapter 7: Lessons learnt

During the development of this project, I was able to demonstrate my knowledge from my time at university. From the client server systems that were implemented in the web application's backend to the penetration testing phase. I have been able to showcase the new technologies available to web developers such as the routing and middleware functionality available with the Laravel framework.

The project started with intensive research activities. This helped me to develop an understanding of the important of research and helped to build my critical skills. It also developed both my technical and non-technical communication skills as well as my presentation skills.

During the development phase of this project, I learned the importance of great time-management skills. In the early stages of this project, I would find myself undertaking multiple objectives which was manageable during the research phase. However, during the development phase this was not manageable, and I learned to better prioritise tasks in order of importance. This helped me to improve my workflow and ensure that the project objectives and other University assignments were met by the given deadlines.

The testing phase of this project highlighted the importance of thoroughly testing the different areas of my work. Regression testing ensured that new changes did not unintentionally affect/break my existing work and taught me the importance of reassessing my work keeping backups. My experience with the usability testing and the feedback I received from volunteers revealed how a small amount of effort can help to promote accessibility for a diverse range of people. Although the project has a focus on security, I learned the importance of being proactive with ensuring security measures are always in place.

Overall, this project was a great learning experience, and it has helped to develop my skills for the cybersecurity industry. I have learned the process of systematically integrating new technologies. Following the Laravel documentation for the database integration and middleware functionality made me understand the importance of the software development lifecycle. Throughout this project I have been able to apply both the practical and theoretical knowledge I have gained from university and online sources.

Chapter 8: Conclusions

8.1 Project Overview

This project had strengthened my professional development in cyber security with secure web development, penetration testing and database security. The project aimed to use industry standard development practices to create a secure web application that will help businesses with their scheduling needs. The primary aims of the project were successfully achieved as the web application is usable on several devices and browsers.

All the main objectives have been met, as well as the optional objectives to research and implement other use cases for this solution. The testing and feedback reflected my goal to produce a secure and accessible scheduling system.

8.3 Legal, social, ethical and professional issues and Acknowledgements.

The website does not infringe on any laws. All user details are protected in the password protected database with encryption for passwords. All images on the website are copyright free, open-sourced or images that I have created.

8.3.1 Acknowledgements

I had drawn inspiration from online sources when implementing the code to style parts of the user interface. This section will acknowledge these sources and provide a reflection on any legal/social/ethical/professional issues.

Scalable Vector Graphics (svgs) were used to style the navbar icons. The XML namespaces were sourced from <http://www.w3.org/2000/svg> to ensure all browsers recognise and correctly display these icons.

The text on the landing page includes HTML, CSS, PHP, Laravel, Burp Suite and Kali Linux for reference only, not to associate any relationship with these entities. Laravel and Burp Suite require prior written permission for their logos, therefore, logos for these companies/entities have not been used.

Code from the following MIT-licensed sources was adapted and only used to style parts of the user interface. A great effort was made to ensure that only sources with the MIT licence were used, accessible at opensource.org/licenses/mit. The MIT licence states: Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files. (OpenSource, n.d) The copyright notice and permission notice are included in the project files, following the industry standard practices.

```

## License

The Laravel framework is open-sourced software licensed under the MIT license(https://opensource.org/licenses/MIT).

## Adapted Code

The following open-sourced code have been adapted, licenced under the MIT license(https://opensource.org/licenses/MIT)
- Navbar layout adapted from https://uiverse.io/akshat-patel28/brown-panther-60, licensed under the MIT License.
- Call-to-action button adapted from https://uiverse.io/akshat-patel28/brown-panther-60, licensed under the MIT License.
- Button animation adapted from https://uiverse.io/arnesinash/cold-turtle-83, licensed under the MIT License.
- FullCalendar Calendar Integration adapted from https://fullcalendar.io/, licensed under the MIT License.

```

Figure 60: MIT Licence notices.

Bibliography

- [1] Apache Friends (2022). XAMPP Installers and Downloads for Apache Friends.
<https://www.apachefriends.org/> [Accessed 14 January 2025]
- [2] Cloudflare Learning (2024, 20 April) Article. What is web application security
<https://cloudflare.com/en-gb/learning/security/what-is-web-application-security/> [Accessed 11 April 2025]
- [3] Feryal Clark MP, (October 2024) Department for Science, Innovation and Technology., Cyber Essentials scheme: overview, <https://www.gov.uk/government/publications/cyber-essentials-scheme-overview>, [Accessed 24 February 2025].
- [4] Flavian, C., Gurrea, R. and Orus, C. (May 2009) Web design: A key factor for the website success.
https://www.researchgate.net/publication/220419456_Web_design_A_key_factor_for_the_website_success. [Accessed 14 January 2025]
- [5] Helmi Mahditia Adam, Widyawan Widyawan and Guntur Dharma Putra (November 2023). A Review of Penetration Testing Frameworks, Tools, and Application Areas.
<https://doi.org/10.1109/icitisee58992.2023.10404397>. [Accessed 11 February 2025]
- [6] Insfran, E., Fernandez, A. (2008). A Systematic Review of Usability Evaluation in Web Development.
https://doi.org/10.1007/978-3-540-85200-1_10. [Accessed 21 March 2025]

- [7] Laravel Documentations, Laravel.com. (2025). Routing - Laravel 12.x - The PHP Framework For Web Artisans.
<https://laravel.com/docs/12.x/routing> [Accessed 25 March. 2025].
- [8] M. A. Jamil, M. Arif, N. S. A. Abubakar and A. Ahmad, (November 2016) Software Testing Techniques: A Literature Review, ICT4M,
DOI: [10.1109/ICT4M.2016.045](https://doi.org/10.1109/ICT4M.2016.045) [Accessed 24 March. 2025]
- [9] Marta F. Arroyabe, Carlos F.A. Arranz, Ignacio Fernandez De Arroyabe, Juan Carlos Fernandez de Arroyabe. (April 2024) Revealing the realities of cybercrime in small and medium enterprises: Understanding fear and taxonomic perspectives, Computers & Security.
<https://doi.org/10.1016/j.cose.2024.103826> [Accessed 12 October 2024]
- [10] Mavituna, F. (2022). SQL Injection Cheat Sheet.
<https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/> [Accessed 22 April 2025]
- [11] NCSC (April 2023). Cyber Essentials: Requirements for IT infrastructure v3.1.
<https://www.ncsc.gov.uk/files/Cyber-Essentials-Requirements-for-Infrastructure-v3-1-April-2023.pdf>
[Accessed 13 January 2025]
- [12] NN Group, (May 2023) Error-Message Guidelines.
<https://www.nngroup.com/articles/error-message-guidelines/> [Accessed 13 December 2025].
- [13] Open Source Initiative, (n.d.) The MIT License.
<https://opensource.org/license/mit> [Accessed 11 December 2025]
- [14] OWASP (2024). *OWASP Top Ten*. <https://owasp.org/www-project-top-ten/> [Accessed 13 January 2025]
- [15] OWASP, (October 2024) Vulnerability Scanning Tools
https://owasp.org/www-community/Vulnerability_Scanning_Tools [Accessed 10 March. 2025]
- [16] Paralect.com. (July 2024). 5 ways to design better white space in your product UI | Particles by Paralect. <https://www.paralect.com/blog/post/5-ways-to-design-better-white-space-in-your-product-ui>
[Accessed 20 March. 2025].
- [17] Prasanth Satya Sai Kiran Gandikota, Deekshitha Valluri, Sathvik Babu Mundru, Gopi Krishna Yanala & S Sushaini. (January 2024) Web Application Security through Comprehensive Vulnerability Assessment.
<https://www.sciencedirect.com/science/article/pii/S187705092302077X#abs0001> [Accessed 03 March 2025]

- [18] Risener, K. (2022, May). A Study of Software Development Methodologies.
<https://scholarworks.uark.edu/cgi/viewcontent.cgi?article=1105&context=csceuh> [Accessed 14 January 2025]
- [19] Shylesh, S. (2017). A Study of Software Development Life Cycle Process Models.
https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2988291 [Accessed 19 April 2025]
- [20] SoftwareSecured, (n.d.) Warren Moynihan. 15 Risks & Rewards of Pentesting in a Production Environment. (n.d.)
<https://www.softwaresecured.com/post/pentesting-in-a-production-environment> [Accessed 13 January 2025]
- [21] S. Kumar, R. Mahajan, N. Kumar and S. K. Khatri, (April 2018) "A study on web application security and detecting security vulnerabilities," 2017 6th International Conference on Reliability
<https://ieeexplore.ieee.org/document/8342469> Doi: 10.1109/ICRITO.2017.8342469 [Accessed 15 January 2025]
- [22] Validator.w3.org., (n.d.) About the Nu Html Checker.
<https://validator.w3.org/nu/about.html> [Accessed 27 March. 2025]