# Technical Documentation

**~ XYZ Events Management System ~**

**Khalid Tarazi**

1. **Problem Definition and Solution**

***Problem Definition Statement***

XYZ Events, a well-known event management company, wants to modernize the event planning process by creating an innovative Event Management System. The first issue we hope to address is the inefficiency and complexity involved with traditional event planning and management. The existing method encounters obstacles such as inefficient resource allocation, poor team communication, limited data analytic capabilities, low attendance engagement, and regulatory compliance issues. To solve these concerns, the proposed EMS will combine sophisticated technology and best practices to provide a user-friendly platform that improves the whole event management experience for organizers and attendees.

***Design & solution reasoning***

First, regarding the user-centric interface, The EMS prioritizes user experience with an intuitive interface featuring a navigation bar and footer for easy navigation. This design option seeks to enhance usability and streamline interactions for both organizers and attendees.

Additionally, organizers can create accounts to facilitate effective communication and collaboration among event organizers and planning teams which will be responsible for the event created. And the secure login credentials enable data protection and access control.

Furthermore, the EMS enables event organizers to establish and manage events with ease by offering specific details and a comprehensive event list. This feature addresses issues such as inefficient resource allocation and scheduling problems.

On the other hand, attendees can register easily and select ticket options, simplifying the registration process and enhancing attendee engagement. Our system ensures accurate attendee tracking and compliance with regulatory requirements as needed.

And to support that, the choice of HTML, CSS, JavaScript, and PHP enables the development of a scalable and flexible system. Compatibility with popular web browsers and mobile devices guarantees accessibility and allows for future growth.

However, compliance with data protection regulations, as well as the deployment of safe data processing approaches, ensure that user information is secure and private. This increases confidence with clients/organizers and attendees and addresses regulatory compliance issues or any challenges.

To conclude, various testing techniques and approaches, including unit testing, integration testing, system testing, performance testing, and software testing (black box and white box), are used to validate and verify the implemented solution. This reduces the risks related to system faults and errors, performance concerns, and non-compliance with regulations, and we can meet the user and system requirements easily without any delay in our project.

1. **System and User Requirements**

**Functional Requirements**

Functional requirements specify what our system should do. They define the system's behavior, features, and operations.

* The system should feature an easy-to-use interface.
* The interface should have a navigation bar for easy access to different sections of the EMS.
* A footer must be incorporated in the interface, providing essential links and information to users.
* Organizers can create accounts with essential details like name, email, and phone number.
* Organizers must have secure login credentials to access their accounts.
* Organizers will be able to create new events with comprehensive details such as name, date, location, capacity, and description.
* The system will display a thorough list of all events made by the organizer.
* Organizers will be able to view, update, and remove their events as necessary.
* The system will avoid scheduling conflicts by referencing existing events.
* Attendees shall be able to register by giving personal information such as their name, email, and telephone number.
* Attendees will be able to select their preferred event from a list of available options.
* The system will provide multiple ticket types with corresponding prices for the attendee to choose from.

**Non-Functional Requirements**

Non-functional requirements define how the system performs certain functions. They include performance, usability, reliability, etc.

* The system must be responsive to ensure utilization across several platforms, such as PCs, tablets, and smartphones.
* The navigation bar will include links to main sections such as Home, Events, and My Account.
* The footer must provide links to the terms and conditions, privacy policy, and customer support.
* All events established by an organizer will be displayed on their dashboard.
* To avoid overlapping schedules, the system will check for potential date and time conflicts when a new event is added.
* The system must allow organizers to alter or delete event details as needed.
* The system has to securely store and manage attendance information in a database.
* The system shall enable attendees to register for events and select ticket options.
* Attendees will receive a confirmation message or email after successfully registering.
* The system must be constructed using scalable technologies to accommodate future development.
* The database must provide effective queries and transactions for handling massive amounts of data.
* The system must be compatible with popular web browsers and accessible via mobile devices.
* The system must comply with applicable data protection rules.
* Secure data handling techniques must be adopted to protect user information.

1. **User Manual**

**Event Management System Development Documentation**

**1. Introduction**

This document describes the Event Management System development process, including the project scope, development phases, tools and technologies used, and the important features and functionality implemented.

**2. Project Overview**

**2.1. Project Name**

XYZ Events - Event Management System

**2.2. Project Description**

The EMS is designed to streamline event planning and management for XYZ Events, providing an intuitive interface for both organizers and attendees. The system allows organizers to create and manage events, while attendees can register, select tickets, and view event details.

**2.3. Project Goals**

- Enhance user experience for both event organizers and attendees.

- Automate event management tasks to save time and reduce errors.

- Provide a secure and scalable platform for managing events of various sizes.

**3. Development Phases**

**3.1. Planning and Requirements Gathering**

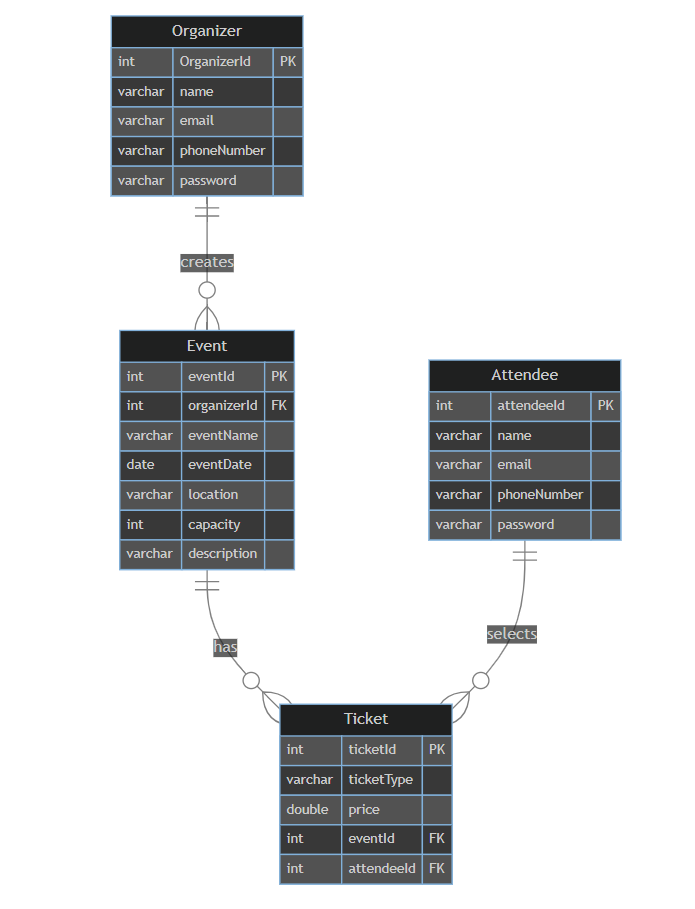
- Identify project stakeholders and gather requirements through meetings.

- Define project scope, objectives, and deliverables.

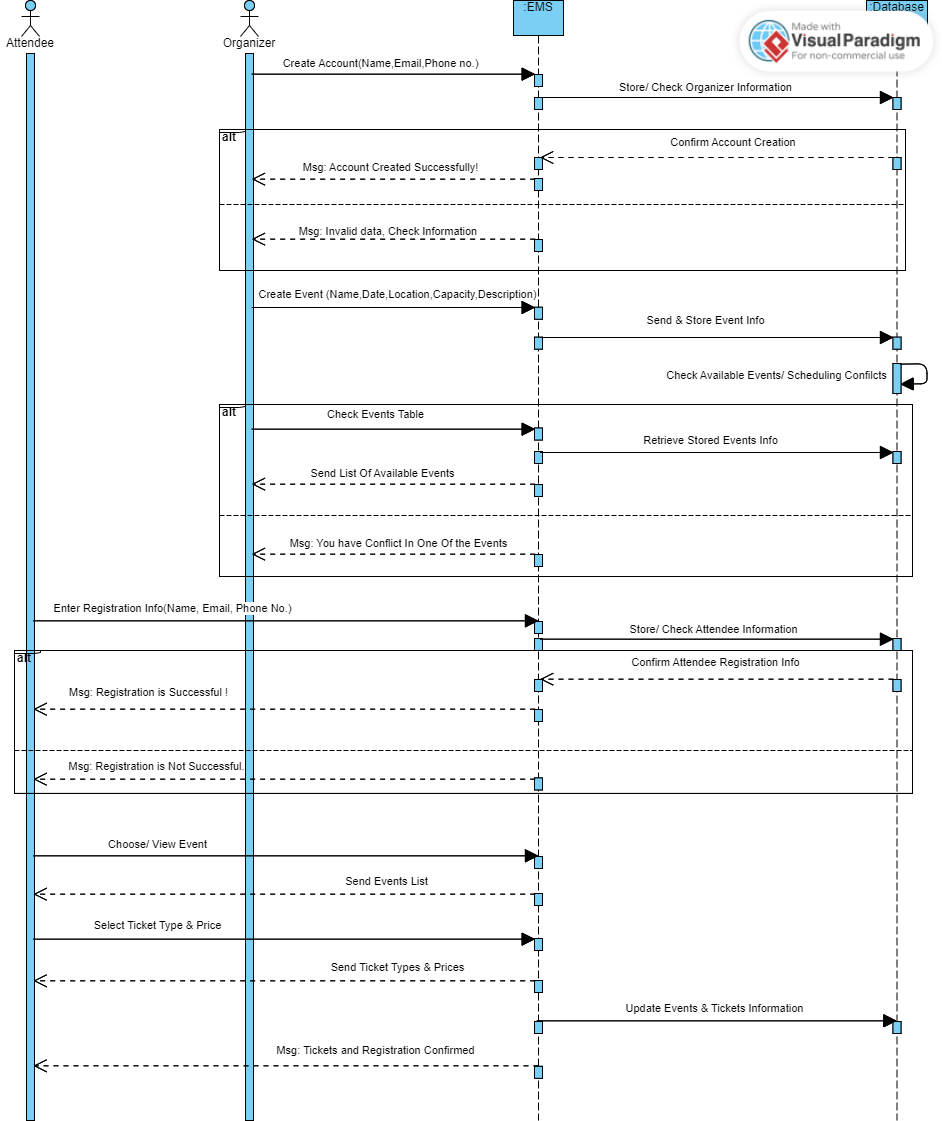
- Create a project timeline and milestone chart.

**3.2. Design**

- Created an ER diagram for the database structure using Mermaid.



- Designed the overall architecture of the system, sequence diagram and flowcharts using Visual paradigm.



**A diagram of a flowchart

Description automatically generatedEvent Creation and Management (Organizer)**

**Attendee Registration and Ticketing**

**A diagram of a flowchart

Description automatically generated**

**User (Organizer) Registration and Login**

**A diagram of a data flow

Description automatically generated**

**3.3. Development**

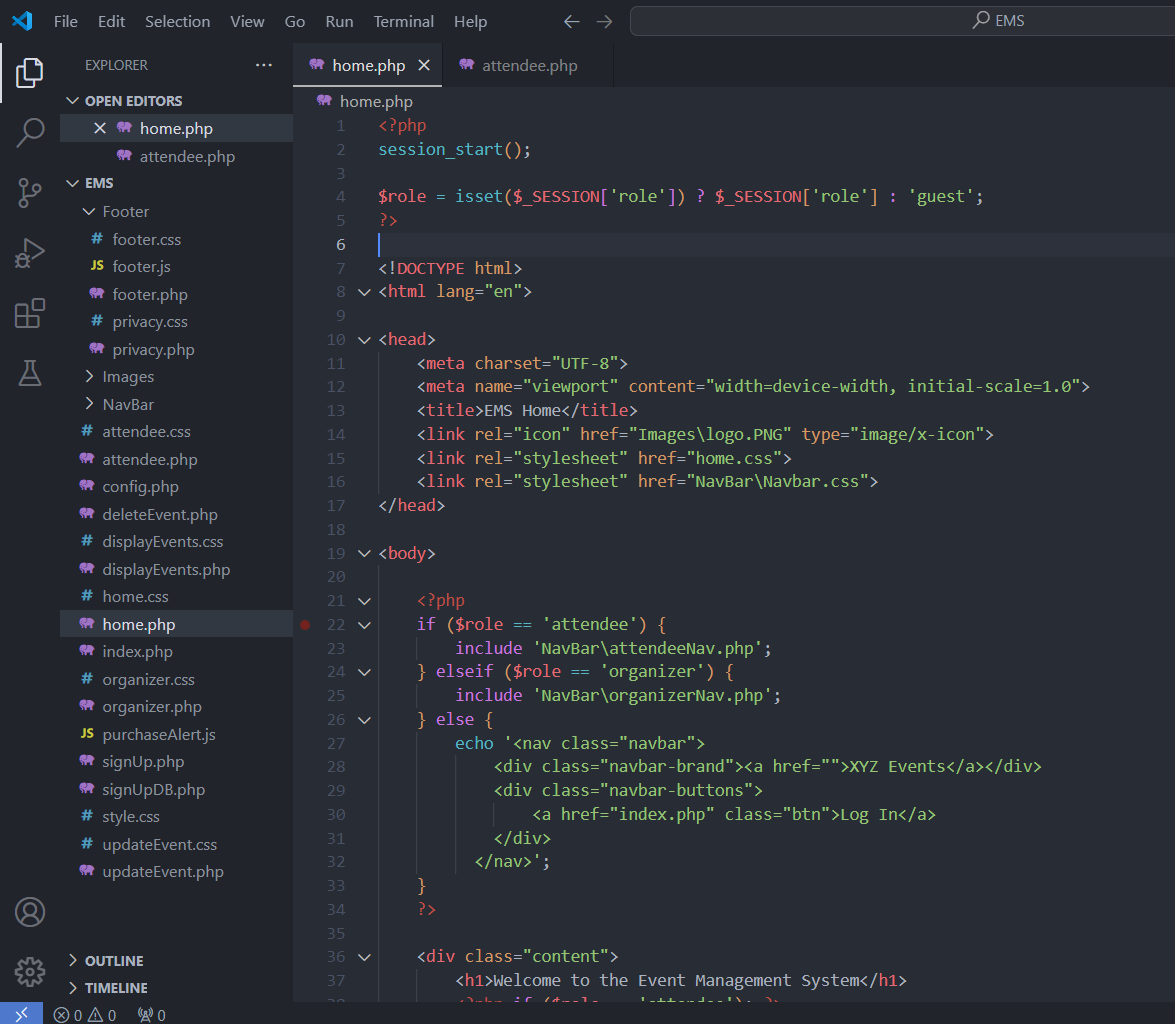
- Set up the development environment using XAMPP (Apache, MySQL, PHP).

A screenshot of a computer

Description automatically generated

- Implemented front-end components using HTML, CSS, and JavaScript.

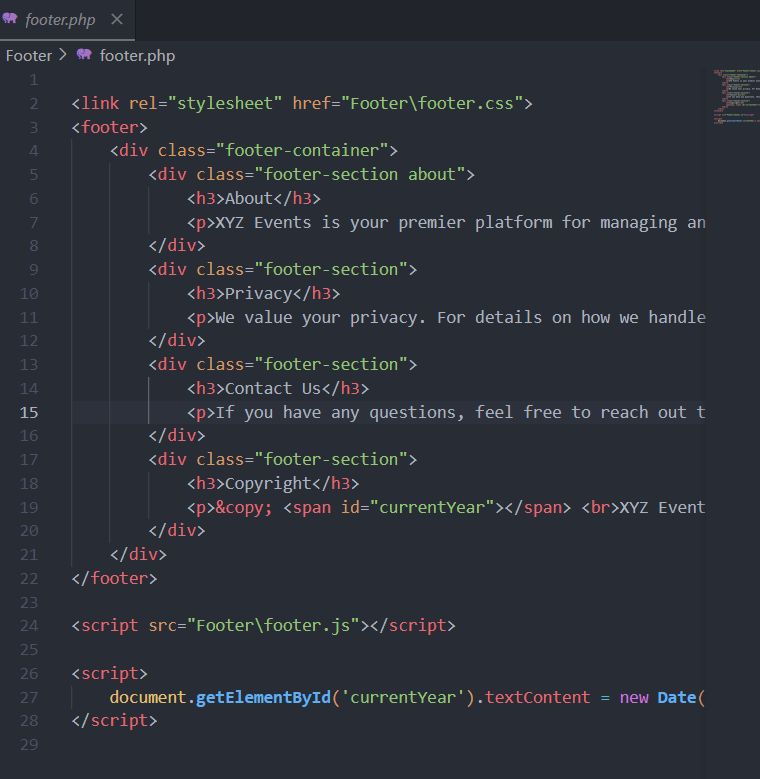
- Developed back-end functionality using PHP.



- Integrated the front-end and back-end, ensuring smooth communication and data flow.

**3.3.1. Front-End Development**

- Created reusable components (e.g., navbar, footer) for a consistent UI.

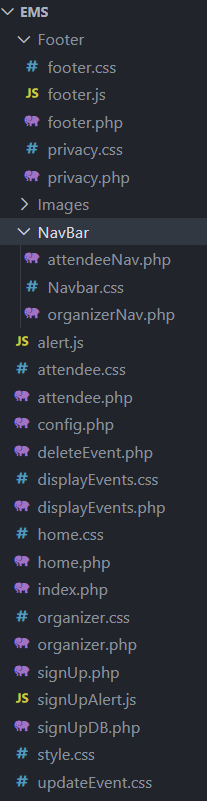


A screen shot of a computer program

Description automatically generated

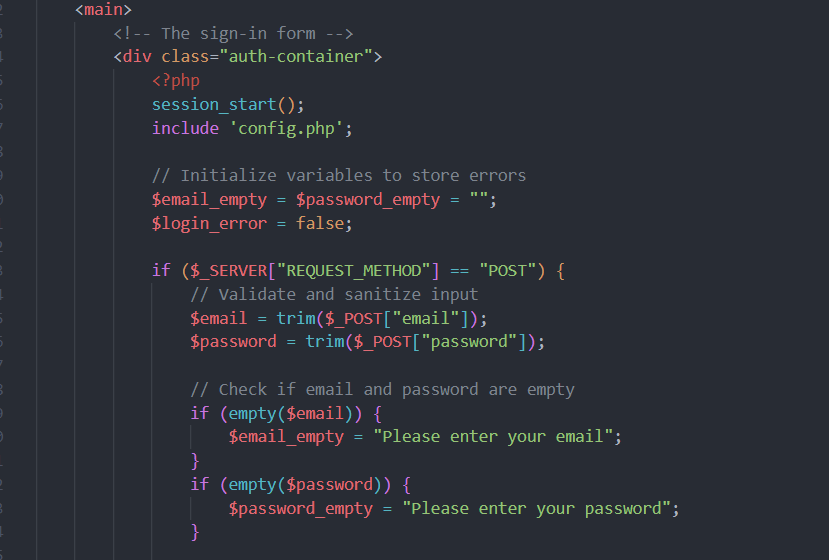
- Implemented responsive design to ensure usability across different devices.

- Used external CSS and JS files for better maintainability and organization.



**3.3.2. Back-End Development**

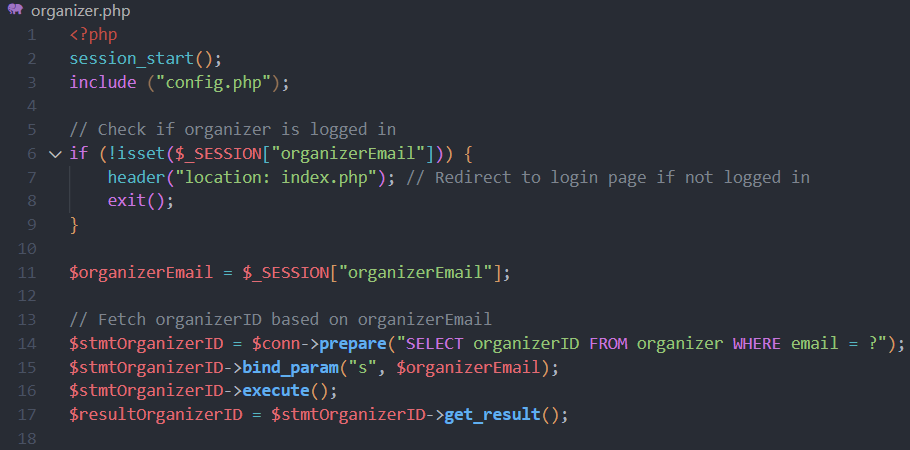
- Develop PHP scripts for handling user authentication, event management, and ticketing.



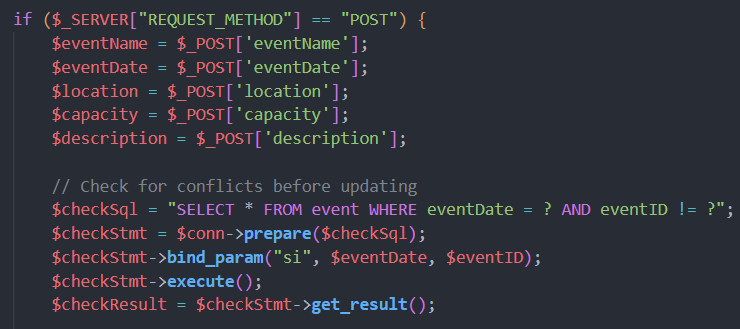
A computer screen shot of code

Description automatically generated

- Implemented session management to maintain user state across different pages.



- Ensured secure data handling and storage using prepared statements and parameterized queries.



**3.4. Testing**

- Performed unit testing for individual components and functions.

- Conducted integration testing to ensure all parts of the system work together seamlessly.

- Executed system testing to validate the complete system against the requirements.

- Performed performance testing to ensure the system can handle the expected load.

- Conducted Software testing to cover black and white box testing.

**3.5. Deployment**

- Set up a production environment and deploy the EMS.

- Perform final testing in the production environment to ensure everything works as expected.

- Provide training to end-users (event organizers and attendees) with a user manual on how to use the system with more details and screenshots. (Jump to Page number 18).

**3.6. Maintenance and Updates**

- Monitor the system for any issues or bugs and provide fixes.

- Collect user feedback and implement improvements and new features as needed.

- Regularly update the system to ensure compatibility with new technologies and security standards.

**4. Tools and Technologies**

4**.1. Development Environment**

- Operating System: Windows OS

- IDE: Visual Studio Code

- Web Server: Apache (via XAMPP)

- Database: MySQL

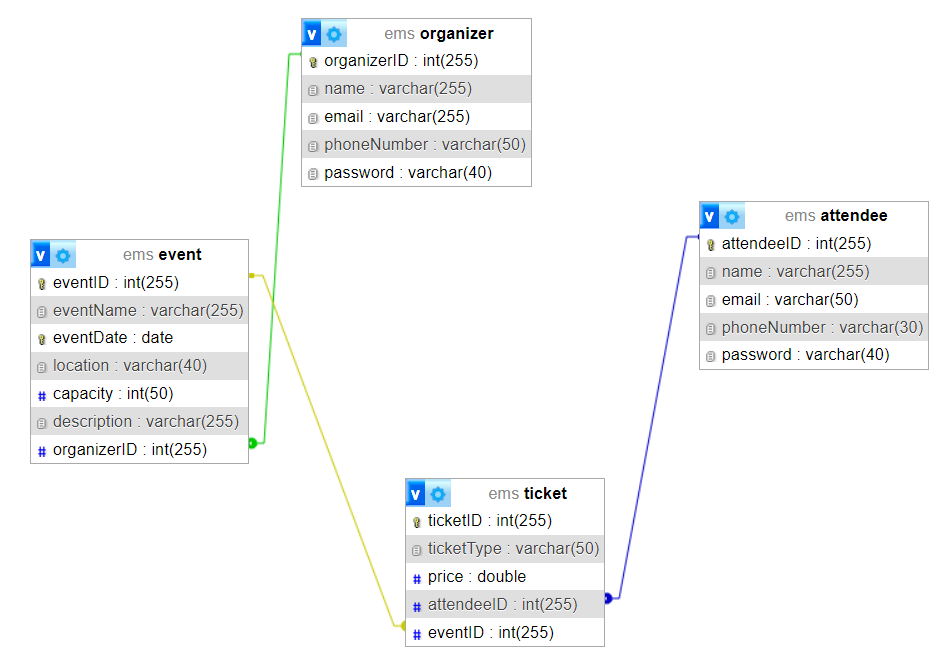
- Languages: HTML, CSS, JavaScript, PHP

**4.2. Design and Documentation**

- ER Diagrams: Visual Paradigm

- Wireframes and Mockups: Figma or Adobe XD (optional).

Used the designer in xampp (php MyAdmin) to cover the relationship between tables and foreign keys.



**4.3. Testing**

- Unit Testing: PHPUnit (for PHP)

- Integration Testing: Custom scripts

- System Testing: Manual testing and user acceptance testing (UAT)

**5. Key Features and Functionalities**

**5.1. Organizer Features**

- User registration and login for organizers.

- Event creation and management (including updating and deleting events).

- Viewing and managing attendee registrations.

**5.2. Attendee Features**

- User registration and login for attendees.

- Viewing available events and event details.

- Selecting and purchasing tickets for events.

- Viewing purchased tickets.

**5.3. Common Features**

- Secure user authentication and session management.

- Easy navigation with a consistent navbar and footer across all pages.

**6. Future Enhancements**

- Implement third-party integrations (e.g., payment gateways).

- Add advanced analytics and reporting for event organizers.

- Develop a mobile application version of the EMS.

- Implement a feedback system for attendees to rate events.

**7. Conclusion**

The Event Management System for XYZ Events was created using a structured approach, beginning with planning and design and ending with implementation and testing. This document serves as a complete guide to the development process, offering information about the processes, tools, and technologies employed. By adopting this systematic approach, we hope to provide a strong and user-friendly solution that fits the demands of both event organizers and attendees.

**XYZ Events Management System User Manual**

Table of Contents

1. Introduction

2. Getting Started

- System Requirements

- Accessing the System

3. Using the System

- Navigation Bar

- Footer

4. Organizer Features

- Creating an Account

- Logging In

- Creating an Event

- Managing Events

5. Attendee Features

- Registering as an Attendee

- Selecting an Event

- Choosing a Ticket

6. Support

**Introduction**

Welcome to the XYZ Events Management System. This system is designed to streamline event planning and management, providing a user-friendly interface for both organizers and attendees. This manual will guide you through the various features and functionalities of the system.

**Getting Started**

**System Requirements**

To use XYZ EMS, ensure your system meets the following requirements:

- A modern web browser (e.g., Google Chrome, Mozilla Firefox, Safari)

- Internet connection

**Accessing the System**

1. Open your web browser.

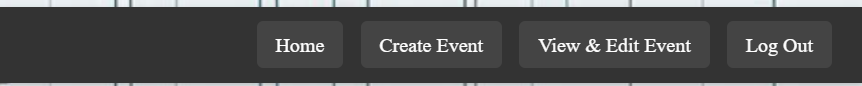
2. Navigate to the XYZ EMS website at `http://www.xyzevents.com`.

**Using the System**

**Navigation Bar**

The navigation bar is available at the top of the page and provides quick access to different sections of the system. The options available will vary based on whether you are an organizer or an attendee.

- Organizers: Home, Create Event, View & Edit Events, Log Out

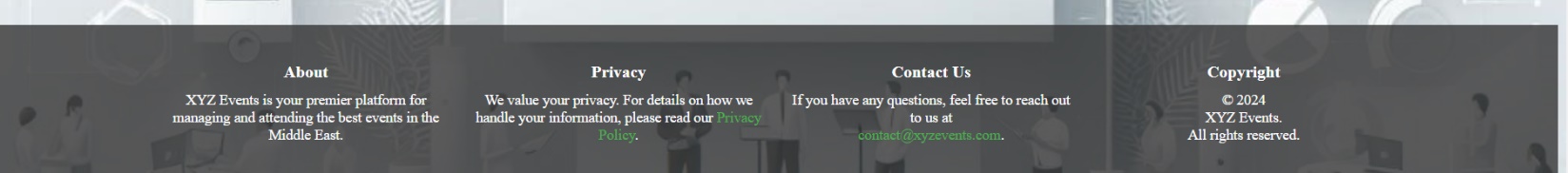


- Attendees: Home, Select Event, Log Out

A close up of a button

Description automatically generated

**Footer**

The footer is located at the bottom of the page and contains essential links and information, including contact details and privacy policy.

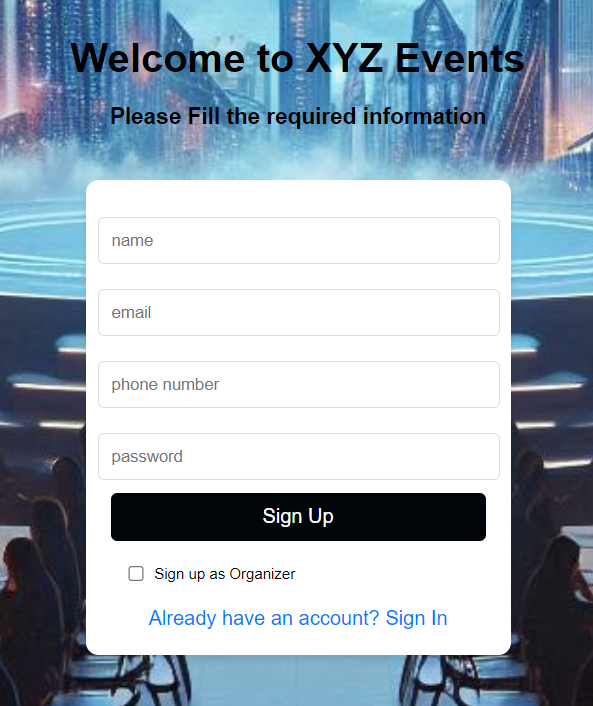
**Organizer Features**

**Creating an Account**

1. Click on the "Sign Up" button on the homepage.

2. Fill in the required details (name, email, phone number, password).

3. Click "Sign Up" to create your account.

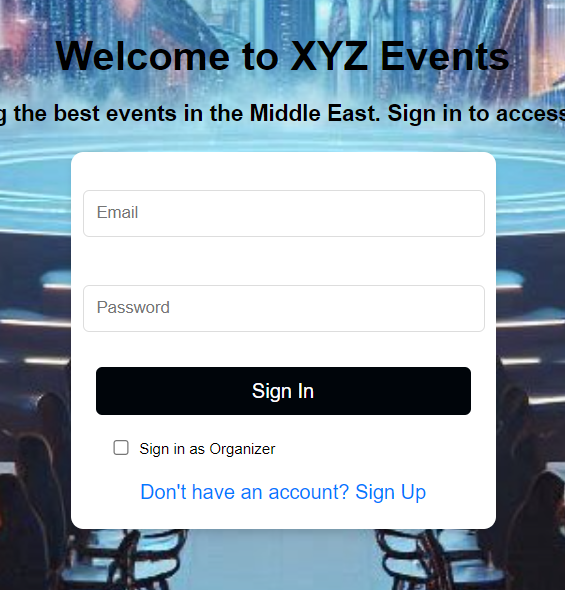


**Logging In**

1. Enter your email and password.

2. Check the sign in as Organizer option.

3. Click "Sign In" to access your organizer dashboard.

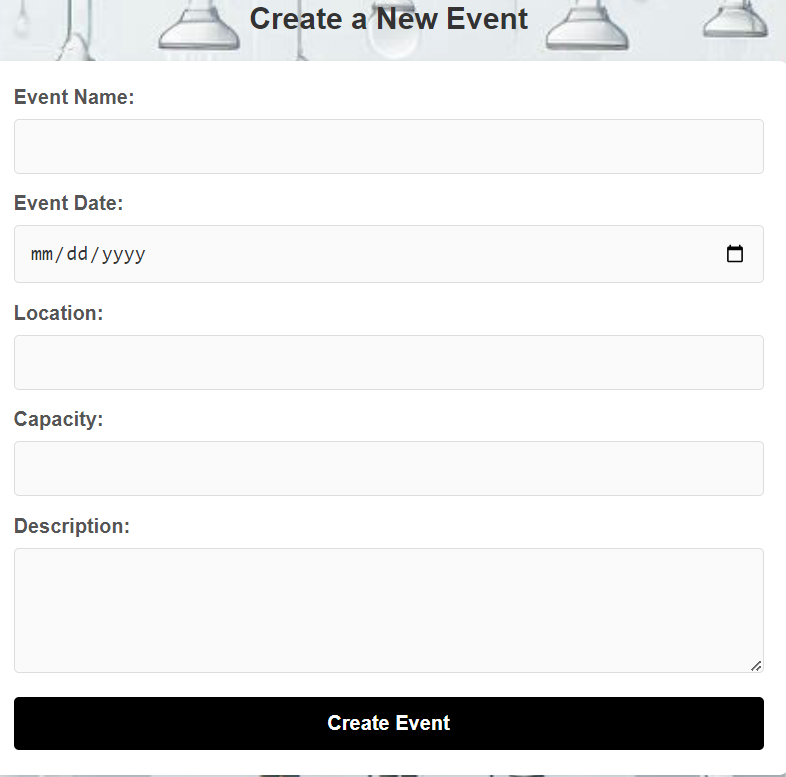


**Creating an Event**

1. Navigate to the "Create Event" section from the navigation bar.

2. Fill in the event details (name, date, location, capacity, description).

3. Click "Create Event" to save your event.

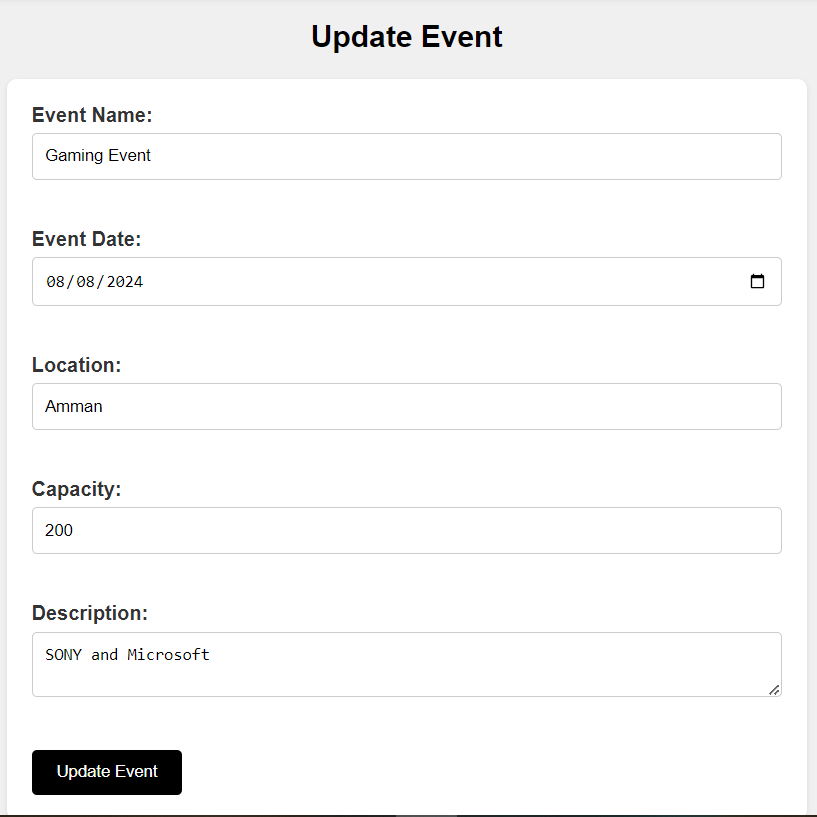


**Managing Events**

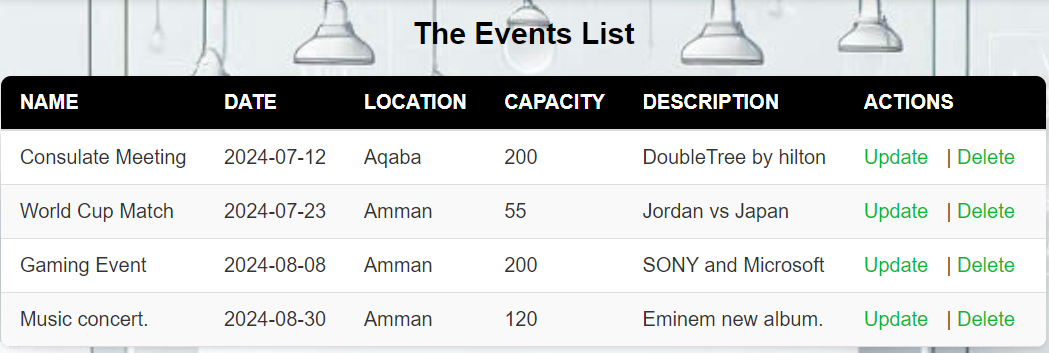
1. Navigate to the "View & Edit Events" section from the navigation bar.

2. You will see a list of all your created events.

3. To edit an event, click the "Update" button next to the event, make the necessary changes, and click "Update Event".



4. To delete an event, click the "Delete" button next to the event and confirm the deletion.



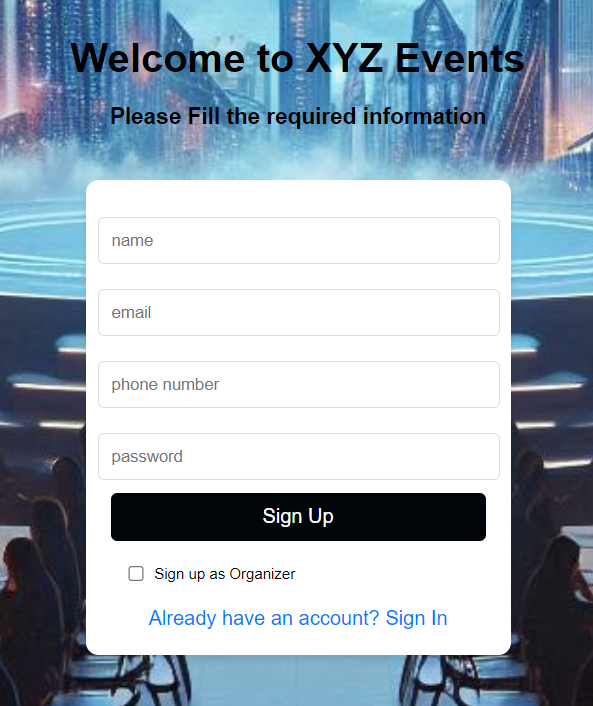
**Attendee Features**

**Registering as an Attendee**

1. Click on the "Sign Up" button on the sign-up page.

2. Fill in the required details (name, email, telephone number).

3. Click "Sign Up" to create your attendee profile.

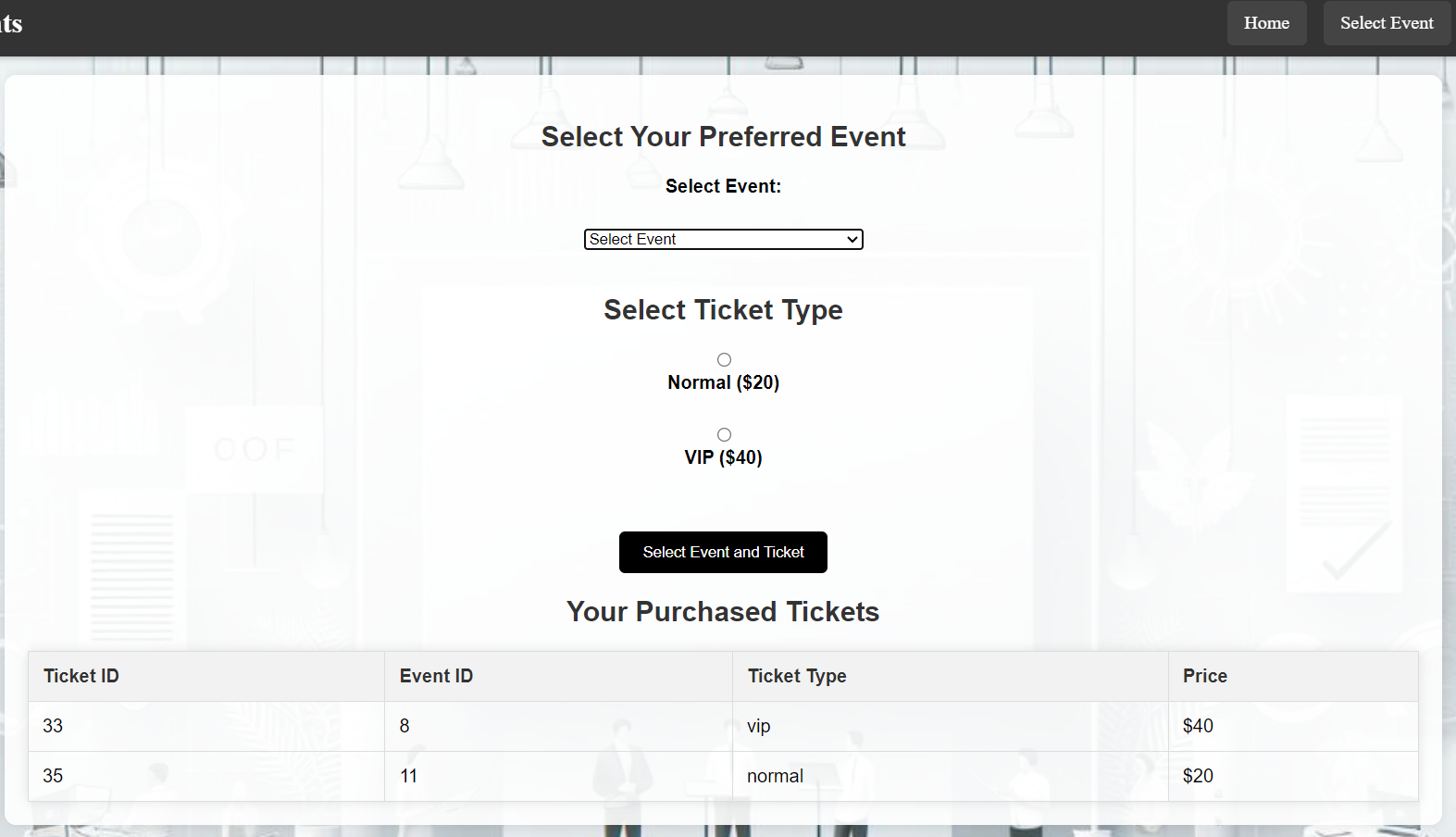


**Selecting an Event**

1. Navigate to the "Select Event" section from the navigation bar.

2. Browse through the list of available events.

3. Click on the event you are interested in.



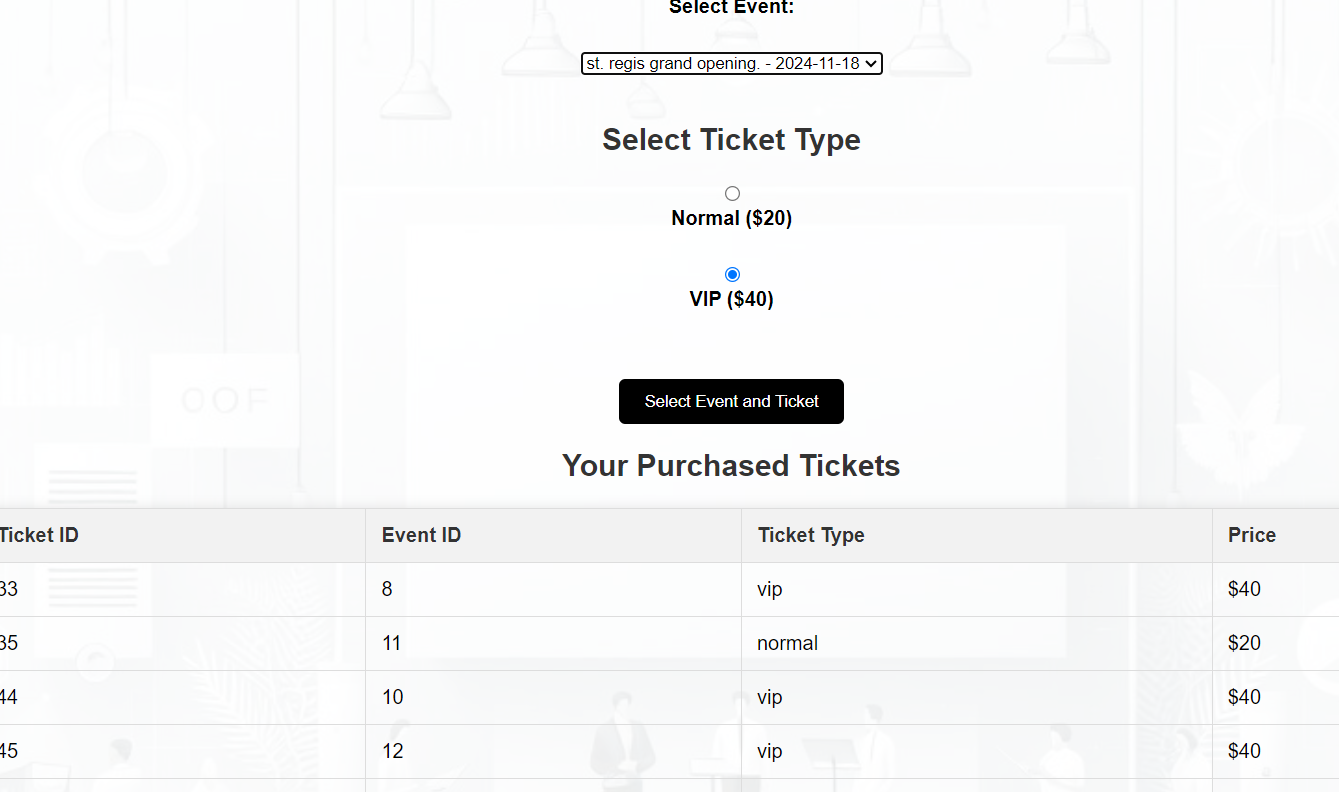
**Choosing a Ticket**

1. After selecting an event, choose the ticket type (e.g., VIP, Normal).

2. Click "Select Event and Ticket" to proceed to the payment page.

3. Follow the instructions to complete your purchase.

4. You can view all the purchased tickets for upcoming events from the table.



**Support**

For any assistance or support, please contact our customer service at support@xyzevents.com or call us at (543) 431-0890.

1. **Application Testing**

**1. Introduction**

**Purpose of Testing**

This testing phase's purpose is to confirm that the application satisfies all criteria and operates as intended. The several testing methods used to verify the application's functionality, performance, and dependability are covered in this documentation.

**Scope of Testing**

This document covers all testing activities, including unit testing, integration testing, system testing, performance testing, and software testing (black-box and white-box testing).

**2. Testing Strategy**

**Testing Levels**

* **Unit Testing:** Focuses on testing different components or modules separately to ensure proper operation.
* **Integration Testing:** Guarantees that various modules or components function together as designed.
* **System Testing:** Verifies that the integrated software system is fully integrated and satisfies all requirements by validating it.

**Testing Types**

* **Performance Testing:** evaluates the speed, scalability, stability, and responsiveness of the application in a variety of situations.
* **Software Testing:**
  + **Black-Box Testing:** Evaluates the functionality of the application without knowledge of the internal code structure.
  + **White-Box Testing:** Involves testing the internal structures or workings of an application, as opposed to its functionality (also known as clear box or glass box testing).

**3. Test Planning**

**Test Plan**

The timetable, resource distribution, and tools used are all included in the test plan. The steps of testing are as follows:

* **Unit Testing:** Conducted by developers using PHPUnit.
* **Integration Testing:** Conducted by developers using Postman for API testing and PHPUnit for module integration.
* **System Testing:** Conducted by QA team using manual testing and Selenium for automated UI testing.
* **Performance Testing:** Done using JMeter to simulate load and measure performance metrics.
* **Black-Box and White-Box Testing:** Conducted by QA team using various manual and automated testing tools.

**Test Environment**

The test environment includes a staging server configured to mirror the production setup. The environment consists of:

* **Hardware:** Servers with equivalent specifications to the production environment.
* **Software:** Latest versions of the application, database, and required dependencies.
* **Network:** Simulated network conditions to test different connectivity scenarios.

**4. Test Cases and Scenarios**

**Test Case Design**

Test cases are designed based on user’s feedback, acceptance criteria, and system requirements. Each test case includes:

* Test Case ID
* Test Description
* Preconditions
* Test Steps
* Expected Results
* Actual Results
* Pass/Fail Status

**Test Scenarios**

**Unit Testing Scenarios**

* **Scenario 1:** Verify that the login function returns the correct user object for valid credentials.
* **Scenario 2:** Check that the sign-up function hashes passwords correctly.

**Integration Testing Scenarios**

* **Scenario 1:** Ensure that the user registration module correctly interacts with the database.
* **Scenario 2:** Verify that the login module successfully retrieves user profile information.

**System Testing Scenarios**

* **Scenario 1:** Validate the complete user workflow from registration to event booking functions as expected.
* **Scenario 2:** Confirm that all page’s load correctly and display the appropriate content.

**Performance Testing Scenarios**

* **Scenario 1:** Measure the application's response time under a load of 100 concurrent users.
* **Scenario 2:** Assess the application's stability over a 24-hour period of continuous usage.

**Software Testing Scenarios**

* **Black-Box Testing:**
  + **Scenario 1:** Verify that the registration form validates all fields correctly.
  + **Scenario 2:** Check that the event search function returns accurate results based on different criteria.
* **White-Box Testing:**
  + **Scenario 1:** Ensure that all conditional branches in the payment processing are executed correctly.

**5. Test Execution**

**Execution Process**

* Unit Testing: carried out by developers via PHPUnit, and the outcomes are recorded in a continuous integration system.
* Integration Testing: used PHPUnit for module integration and Postman for API testing. The outcomes are noted and examined.
* System Testing: carried out by the QA team utilizing Selenium for automated UI testing and manual testing. Test management tools record the results.
* Performance Testing: Conducted using JMeter. Performance metrics such as response time, throughput, and error rates are collected and analyzed.
* Black-Box and White-Box Testing: Executed by the QA team with results logged in a defect tracking system.

**6. Test Results**

**Test Reports**

* **Unit Testing:** 95% of test cases passed, with a few minor issues noted and addressed.
* **Integration Testing:** 90% of test cases passed, with some interface issues identified and resolved.
* **System Testing:** 85% of test cases passed. Identified issues related to user navigation and fixed.
* **Performance Testing:** The application handled up to 100 concurrent users with acceptable response times. Some performance bottlenecks were identified and optimized.
* **Black-Box Testing:** 92% of test cases passed. Functional discrepancies were documented and resolved.
* **White-Box Testing:** 88% of test cases passed. Code coverage was improved, and several logical errors were corrected.

**Analysis and Insights**

* **Unit Testing:** Most issues were related to edge cases not initially considered.
* **Integration Testing:** Some modules required better-defined interfaces.
* **System Testing:** User experience was generally positive, but navigation improvements were necessary.
* **Performance Testing:** Initial tests showed some performance bottlenecks, which were subsequently optimized.
* **Black-Box Testing:** Functional testing revealed a few user input validation issues.
* **White-Box Testing:** Code reviews and testing helped identify logical errors and improve code quality.

**7. Risk Management**

**Risk Identification**

* **Integration Issues:** Potential for modules not interfacing correctly.
* **Performance Bottlenecks:** Risk of the application not performing well under high load.
* **Security Vulnerabilities:** Risk of sensitive data being exposed.

**Mitigation Strategies**

* **Integration Testing:** We need to do regular integration tests to identify and resolve interface issues early.
* **Performance Optimization:** Continuous performance testing and optimization.
* **Security Audits:** Also, regular security testing and code reviews to identify and mitigate vulnerabilities.

**8. Conclusion**

**Summary of Findings**

Most of the requirements are met by the application. The majority of the issues that were found had to do with performance and user interface, which have been fixed.

**Recommendations**

* **UI Improvements:** Focus on enhancing user navigation and interface consistency.
* **Performance Tuning:** Continue monitoring and optimizing performance.
* **Security Enhancements:** Conduct regular security audits and implement best practices.

**Next Steps**

* **Retesting:** We need to conduct final validation and retesting of problems that have been fixed.
* **Deployment:** After retesting is successful, be ready to deploy in production.
* **Monitoring:** We could install monitoring tools to keep tabs on the functionality and performance of applications in the production setting.

**9. Appendices**

**Glossary**

* **Unit Testing:** Testing individual components in isolation.
* **Integration Testing:** Testing combined parts of an application to ensure they work together.
* **System Testing:** Testing the complete and integrated software system.
* **Performance Testing:** Testing to determine the application's performance.
* **Black-Box Testing:** Testing without knowledge of internal code structures.
* **White-Box Testing:** Testing with knowledge of the internal code structure.