King Saud University
College of Computer and Information Sciences
Computer Science Department

"Eventia: Events Management Platform"

CSC 497 – Final Report

Prepared by:

Abdulrahman Alghanmi	441102300
Abdullah Alshammari	443100803
Turki Almuayli	443102104
Abdullah Binradyan	443101250

Supervised by:

Dr. Mohammed Alsulmi

Software project for the degree of Bachelor in Computer Science

Second Semester 1447/1448

Spring 2026

I.Acknowledgements

We would like to express our sincere gratitude to King Saud University and the College of Computer and Information Sciences for their continuous support throughout this project. Our special thanks go to Dr. Mohammad Alsulmi for his valuable guidance, feedback, and encouragement. We are also deeply thankful to our colleagues and peers for their cooperation and motivation, which greatly contributed to the success of this project.

II.English Abstract

Event organizers often face complex challenges in coordinating between multiple stakeholders, including government authorities, service providers, vendors, sponsors, and attendees. Traditional approaches rely on paper-based processes, emails, and scattered communication methods that are time consuming, likely to cause mistakes, and inefficiency. These limitations slow down event preparation and reduce the overall quality of the experience delivered to attendees.

Eventia is an events management system designed as a centralized platform that handles the entire event lifecycle. It integrates essential features such as event creation and scheduling, ticket booking, vendors invitations and collaborations. It also provides post event analytical reports including attendees' feedbacks and surveys to the organizers in order to evaluate the performance and guide future improvements.

What distinguishes Eventia is its innovative approach to vendor and stakeholder collaboration. The system expands the concept of vendors beyond traditional service providers to include logistic services providers too such as government authorities, security agencies, restaurants, sponsors, and other entities that play a crucial role in successful event execution. Through Eventia, organizers can directly submit service requests, track their status, and receive approvals digitally in a fraction of the time compared to conventional methods such as paperwork or email correspondence.

By combining event management tools with a unified communication channel for all stakeholders, Eventia ensures efficiency, transparency, and flexibility in organizing events. Attendees benefit from seamless registration including all required information, vendors and authorities experience simplified collaboration, and organizers gain the ability to focus on delivering exceptional event experiences. Ultimately, Eventia transforms event management into a smarter, faster, and more organized process, setting a new standard for how events are planned, executed, and evaluated.

III.Arabic Abstract

يواجه منظمو الفعاليات تحديات في الربط بين أطراف متعددة من أصحاب المصلحة تشمل الجهات الحكومية، ومقدمي الخدمات، والرعاة، والحضور. وتعتمد الأساليب التقليدية على الإجراءات الورقية، والبريد الإلكتروني، ووسائل الاتصال الأخرى، وهذه الأساليب تستغرق وقتًا طويلًا، وتزيد من احتمالية وقوع الأخطاء، وتفتقر إلى الكفاءة. مما يؤدي إلى بطء في مرحلة التحضير للفعاليات وتؤثر سلبًا في جودة التجربة المقدمة للحضور.

إيفينتيا هو نظام لإدارة الفعاليات صُمّم ليكون منصة مركزية تُبسط دورة حياة الفعالية بأكملها. يدمج النظام خصائص أساسية مثل: إنشاء الفعاليات وجدولتها، حجز التذاكر، دعوة المورّدين والتعاون معهم. توفر المنصة أيضاً تقارير تحليلية بعد الفعالية تتضمن الاستبانات والتغذية الراجعة من حضور الفعالية بهدف تقييم الأداء وتوجيه التحسينات المستقبلية.

ما يميز إيفينتيا هو النهج المبتكر في التعاون بين مزودي الخدمات وأصحاب المصلحة. إذ يوسع مفهوم مزودي الخدمات ليشمل الخدمات اللوجستية إلى جانب مقدمي الخدمات التقليديين مثل الجهات الحكومية، والجهات الأمنية، والمطاعم، والرعاة، وغيرهم من الكيانات التي تساهم في نجاح الفعاليات. ومن خلال النظام، يمكن للمنظمين ومزودي الخدمات تقديم طلبات مباشرة، وتتبع حالتها، والحصول على الموافقات رقميًا في فترة زمنية أقصر بكثير مقارنة بالأساليب التقليدية كالمراسلات الورقية أو البريدية.

وبالجمع بين أدوات إدارة الفعاليات وقناة اتصال موحدة لجميع الأطراف المعنية، تتضمن منصة إيفينتيا تحقيق الكفاءة والشفافية والمرونة في تنظيم الفعاليات. من خلال ذلك، يستفيد الحضور من عملية تسجيل سلسة تتضمن جميع المعلومات المطلوبة، ويحظى المورّدون والسلطات بتجربة تعاون مبسطة، بينما يكتسب المنظمون القدرة على التركيز في تقديم تجارب استثنائية. وبهذا، يُعيد إيفينتيا تشكيل عملية إدارة الفعاليات لتصبح أكثر ذكاءً وسرعةً وتنظيماً، وتكوين معيارًا جديدًا لكيفية التخطيط والتنفيذ والتقييم في هذا المجال.

Table of Contents

Table of Figures

Table of Tables

Chapter 1: Introduction

Organizing events often involves coordinating between multiple stakeholders such as government authorities, service providers, vendors, sponsors, and attendees. Traditional methods like paperwork, emails, and scattered communication channels are inefficient, prone to errors, and time-consuming, which slows down preparation and negatively impacts the attendee experience. These limitations highlight the need for a modern solution that streamlines communication, planning, and execution across all parties involved.

Eventia is designed as a centralized events management system that supports the entire event lifecycle from planning and scheduling to ticketing, collaboration, and post-event analysis. By expanding the concept of vendors to include logistics providers, government agencies, and security authorities, Eventia fosters efficient collaboration and faster approvals through a unified platform. This approach ensures transparency, flexibility, and efficiency for all stakeholders while giving organizers the tools to deliver high-quality experiences. Ultimately, Eventia transforms event management into a more efficient and organized process, setting a new standard for how events are executed and evaluated.

1.1 Problem Statement

Managing events has become increasingly difficult as the number of stakeholders and requirements continues to grow. Organizers must handle government approvals, vendor coordination, sponsorship agreements, security arrangements, and attendee management often all at the same time. Relying on scattered tools like paper forms, emails, and phone calls leads to confusion, delays, and a higher risk of mistakes. These inefficiencies waste time and resources, while also damaging the reputation of events when attendees experience poor organization, delays, or missing services. Without a centralized system, critical tasks such as approvals, scheduling, and communication remain slow, fragmented, and unreliable.

A centralized system directly address these challenges by providing a unified platform that transforms event management into a more efficient, organized, and professional process.

1.2 Goals and Objectives

The goal of this project is to design a centralized web-based event management system that simplifies event organization, enhances collaboration, and improves overall efficiency for organizers, vendors, sponsors, and attendees. The objectives to achieve this include:

- Develop a responsive web platform for event creation and scheduling.
- Implement a role-based access control system with personalized dashboards.
- Establish a direct communication gateway between event organizers and the Saudi Conference & Exhibition General Authority (SCEGA) to obtain necessary approvals and ensure full compliance with the Kingdom’s regulations.

- Establish a vendor and stakeholder collaboration module with digital invitations and approvals.
- Build a ticket booking and attendee management system.
- Provide post-event feedback tools and analytical dashboards for performance evaluation.

1.3 Solution

Eventia is a web-based event management platform designed to centralize and simplify the entire event lifecycle. It provides organizers, vendors, sponsors, and authorities with a single digital space to coordinate tasks, manage approvals, and track progress efficiently.

By replacing scattered tools such as emails and paper forms. Eventia reduces delays, minimizes errors, and strengthens communication between all parties.

Eventia transforms event management into an efficient, transparent, and professional process. It ensures smoother operations for organizers, enables easier collaboration with vendors and authorities, and delivers better overall experience for attendees.

1.4 Research Scope

The scope of this project extends to the design and development of a comprehensive event management platform intended primarily for the Saudi Conventions & Exhibitions General Authority (SCEGA). Beyond this core audience, the platform can also serve other government agencies and private organizations who manage events within the Kingdom of Saudi Arabia. In addition, the system serves individual users by allowing them to register, view event details, and engage in events directly. The scope further encompasses sponsors, vendors and services providers, who play a critical role in the success of events, by providing them with dedicated features for collaboration requests and participation. However, the system does not generate or manage event-related content such as promotional materials, presentations, or media coverage, as content creation remains the responsibility of event organizers. Additionally, the platform is not intended to support events outside the Kingdom of Saudi Arabia, with its features and compliance requirements designed specifically for domestic events.

Chapter 2: Background

The development of a centralized event management platform relies on established concepts in web development, including front-end design, back-end logic, data storage, and system integration. This chapter introduces the background of various potential tools and technologies and explains how they provide the foundation for creating modern web-based platforms with efficient and robust designs. Coming up, we'll dive into front-end and back-end development, take a look at databases, and unpack how APIs tie everything together.

2.1 Front-end Development

Front-end development refers to designing and building the user interface with the help of markup languages and various tools. It focuses on the client side of an application, where user interaction takes place. This involves creating the visible parts of a website or app, such as buttons, text, layouts, and other elements that users interact with. It also makes use of different programming languages and frameworks. [1]

2.1.1 HTML

HyperText Markup Language abbreviated as HTML is the standard language in which Web pages are structured and created. It determines the structure of a webpage in terms of a series of elements that direct the browser on the way content should be shown. These act as labels for the different sections in the page like headings, paragraphs, pictures, and links such that the information is arranged correctly and formatted. [2]

Figure 1: HTML5 logo [48]

HTML is a supporting technology in the creation of structured web content and was introduced in 1993 and is now at version HTML5. It enables developers to develop forms, tables, and other structured data setups needed in the management of input and the presentation of information. As the web has expanded over time, HTML has played a pivotal role in enhancing it to support interactive, accessible and cross platform experiences on a broad spectrum of devices. [1]

2.1.2 CSS

CSS, short for Cascading Style Sheets, is a language used to design and control the presentation of web pages. It determines how HTML elements are displayed across different media, such as screens. With CSS, developers can efficiently manage the layout and appearance of multiple web pages, reducing repetition and improving consistency. Styles can also be stored in external CSS files, making them easier to organize and reuse. [3]

Figure 2: *CSS Logo [48]*

Introduced on December 17, 1996, and now at version CSS3, CSS plays a key role in modern web development. It goes beyond basic styling by allowing control over colors, layouts, and responsive designs that adapt content smoothly to different screen sizes and devices. These features ensure that websites remain visually appealing, consistent, and user-friendly across all platforms. [1]

2.1.3 JavaScript

JavaScript was created in order to provide web pages with interactivity and dynamism. The code which is known as scripts can be directly incorporated into an HTML document and it automatically runs when the page is loaded. Scripts are not compiled or pre-configured, and this is a difference between JavaScript and programming languages such as Java. [4]

***Figure 3**: JavaScript Logo [49]*

It is the basis of JavaScript, having first been standardized as ECMAScript on December 4, 1995, and is nowadays known as ECMAScript 2022. ECMAScript is a single threaded, cross-platform, and lightweight programming language that is dynamic in behavior, and event-oriented. JavaScript has the following features that allow it to be used in the creation of responsive and interactive web applications. [1]

2.1.4 Bootstrap

Bootstrap is an open-source front-end framework, which includes a full set of HTML, CSS, and JavaScript components, released in August 2011 and is currently on version 5.3.2. It has a responsive grid system, pre-built interface components, and a collection of CSS and JavaScript utilities, and customizable themes and templates that can be used to accelerate web development. [1]

***Figure 4**: Bootstrap logo [72]*

Bootstrap is one of the most used frameworks as a method of creating responsive mobile-first websites. It also has many ready-made UI elements like buttons, modals, and navigation bars in addition to its core components. The framework also has in-built support of responsive typography, spacing and utility classes, which guarantee uniform and flexible layouts across devices. Also, Bootstrap can be highly customized by using Sass variables and configuration options, which means that their web interfaces can be made to meet the requirements of a particular project. [5]

2.1.5 ReactJS

ReactJS is a free, open-source JavaScript library used to build dynamic and interactive user interfaces. Known for its simplicity, efficiency, and component-based structure, React allows developers to build applications using reusable, independent blocks of code. This modular approach has made React one of the most popular tools in modern web development.

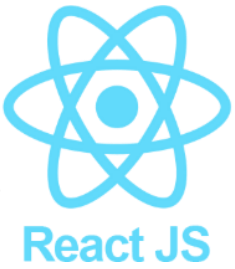

***Figure 5**: ReactJS [50]*

Whether you're working on a small project or a large-scale application, React provides a lightweight yet powerful toolkit for quickly building functional web apps. In addition, a vibrant community of developers contributes prebuilt components and advanced tools—such as state management and routing that help streamline the development process and boost productivity. [6] [1]

2.1.6 Angular

Angular is a powerful web framework which is meant to assist the developers create effective and high performing applications. Supported by Google, it has a broad collection of tools, libraries, and APIs that make developing and being more productive easier. Angular has a robust base on building applications that are fast, reliable and at the same time are easy to scale when teams and projects are expanded.

***Figure 6**: Angular Logo [51]*

Angular (started on October 20, 2010) is a free software architecture consisting of JavaScript using the MVC framework, is an open source project. It is applied in building single-page applications (SPAs) that provide the fast page delivery and the modular and structured framework of the modern web development. [7]

2.1.7 jQuery

jQuery is a lightweight, high-performance, open-source JavaScript library introduced in August 2006. Currently at version 3.7.1, it simplifies tasks like manipulating the DOM, handling events, creating animations, managing AJAX interactions, and building dynamic web experiences all while supporting cross-browser compatibility. [1]

***Figure 7**: JQuery [52]*

With its user-friendly API, jQuery streamlines complex JavaScript operations, making code more maintainable, readable, and efficient. This ease of use has made it one of the most widely adopted tools in web development. It enables

developers to build rich, responsive user interfaces quickly while ensuring consistent behavior across platforms and browsers. [8] [9]

2.2 Back-end Development

The Server is the back end aspect of a Web application as it is concerned with logic, database and procedures that maintain a seamless operation of web servers. It contains the basic functionality of the application, and the server manages the interaction with the database where the operations required to provide functionality to the frontend are carried out. This layer deals with the functionality and vital aspects of the application, data processing, and the general functionality. Some of the common backend technologies are node.js in the case of JavaScript, Django in the case of Python and Spring in the case of Java.

2.2.1 JavaScript & NodeJS

***Figure 8**: NodeJS Logo [53]*

Node.js is a cross-platform, open-source runtime that allows developers to run JavaScript outside the browser, making it a versatile choice for everything from building APIs to powering real-time applications. Instead of using traditional multi-threading, Node.js runs on a single-threaded architecture with asynchronous, non-blocking I/O. That means tasks like reading from a file, accessing a database, or making a network request won't freeze up the entire application. Its event-driven model is one of its standout features, enabling a single server to manage thousands of concurrent connections efficiently. Another big advantage is its seamless integration with frontend development since it's all JavaScript, developers can work across the stack without switching languages. Plus, Node.js keeps pace with modern JavaScript standards, so accessing new ECMAScript features is often just a matter of updating your Node version or enabling a few flags. [10]

2.2.2 Python & Django

***Figure 9**: Django Logo [55]*

Django is a powerful, open-source web framework written in Python, built to take the hassle out of backend development. It handles much of the behind-the-scenes work that web apps typically require, so developers can concentrate on building features instead of reinventing the basics. Rooted in the DRY (Don't Repeat Yourself) philosophy, Django promotes clean, reusable code and helps speed up development without sacrificing structure. It comes with essential tools

already built in like user authentication, database support, and full CRUD (Create, Read, Update, Delete) functionality making it especially useful for projects that rely on managing data. With Django doing the heavy lifting, developers are free to focus on writing better logic and creating a smoother experience for users. [11]

2.2.3 PHP & Laravel

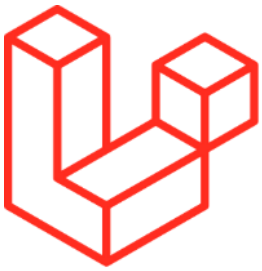

***Figure 10**: PHP Logo [56]*

PHP (short for Hypertext Preprocessor) is a popular and flexible scripting language that plays a major role in web development. Known for its speed and reliability, PHP powers everything from personal blogs to large-scale web platforms. Built on top of PHP, Laravel is a modern, open source framework designed to make web application development faster and more manageable. It follows the Model-View-Controller (MVC) design pattern, which helps break the backend into clear, organized parts making it easier to build, scale, and maintain applications over time. [12]

2.2.4 C# & .NET

***Figure 11**: .Net Logo [54]*

C# (pronounced C-sharp) is a modern, object-oriented programming language developed by Microsoft in 2000 as part of the .NET ecosystem. It's widely used to build Windows applications, web services, and a variety of software solutions. C# blends the power of C/C++ with the simplicity and structure of languages like Java and Visual Basic, making it both efficient and approachable. [13]

The .NET Framework, also created by Microsoft, is a development platform that offers a runtime environment, rich libraries, and essential tools for building and running applications on Windows. While the original .NET was Windows-focused, the release of .NET Core now simply called .NET starting with version 5 brought true cross-platform support. Today, developers can use languages like C#, F#, and VB.NET to build apps for the web, desktop, mobile, cloud, and even gaming, across multiple operating systems. [14]

2.2.5 Ruby & Ruby on Rails

Ruby is a dynamic, open-source programming language designed with a strong focus on simplicity and developer happiness. Its clean, easy-to-read syntax makes writing code feel natural and enjoyable, especially for those who value elegance in software design. [15]

***Figure 12**: Rails [57]*

Built on Ruby, Rails short for Ruby on Rails is a full-stack web framework that comes packed with everything you need to build modern web applications. It takes care of essential features like rendering HTML, managing databases, sending emails, running background jobs, integrating real time functionality with WebSockets, handling file uploads to the cloud, and guarding against common security threats. The combination of Ruby and Rails offers a streamlined, efficient approach to web development that is both powerful and maintainable over the long term. [16]

2.2.6 Java & Spring Boot

Spring Boot is a framework built to make developing web applications and microservices with Java faster and easier. It builds on the core Spring Framework a widely used, open-source toolkit for creating robust, production ready applications that run on the Java Virtual Machine (JVM). What sets Spring Boot apart is its focus on simplicity. It reduces setup time by offering automatic configuration, embracing a convention-over-configuration mindset, and supporting stand-alone applications that can run with minimal boilerplate. This streamlined approach helps developers get Spring projects up and running quickly, without getting bogged down in complex setup. For added flexibility, Spring Boot applications can also be fine-tuned and deployed using the Open Liberty runtime, making them even more adaptable to various deployment environments. [17]

***Figure 13**: Spring Logo [58]*

2.3Databases

Databases are organized collections of data that enable efficient storage, retrieval, and management of information. They are an important part of computing in the modern world since they provide consistency, security, and extensibility of data. Databases are the center of nearly every computer system, such as web applications, financial systems, healthcare records, social networks, and mobile apps. They can be broadly categorized into relational databases that make use of structured schemas and SQL, and NoSQL databases

that are designed for scalability and flexibility with unstructured or semi-structured data. [18] [19]

2.3.1 MySQL

***Figure 14**: MySQL Logo [59]*

MySQL, first developed by MySQL AB in 1995 and acquired by Oracle Corporation in 2010, is a widely used open source relational database system. Known for its speed, reliability, and ease of use, MySQL follows standard SQL and remains a popular choice for powering web applications. [20]

Its strong community support and wide range of tools make it a go-to solution for developers across industries. MySQL is trusted by high traffic platforms like Facebook and YouTube, and it's also commonly used in banking systems, e-commerce platforms, and content management systems such as WordPress and Drupal. [21]

2.3.2 Microsoft SQL Server

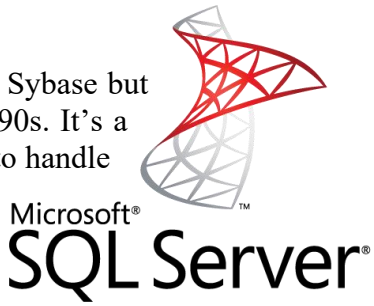

***Figure 15**: Microsoft SQL logo [60]*

Microsoft SQL Server began as a joint project with Sybase but became fully owned and developed by Microsoft in the 1990s. It's a powerful relational database management system designed to handle everything from online transaction processing (OLTP) to complex business intelligence tasks all tightly integrated within the Microsoft ecosystem. [22]

It's widely used across industries like retail, finance, and healthcare to manage critical business data. With built in support for advanced analytics and reporting through tools such as SQL Server Analysis Services and SQL Server Reporting Services SQL Server has established itself as a strong solution for organizations seeking robust business intelligence capabilities. [23]

2.3.3 Firebase

***Figure 16**: Firebase logo [24]*

Firebase was originally launched by Firebase Inc. in 2011 as a Backend as a Service (BaaS) platform and was acquired by Google in 2014. One of its core features is the Firebase Realtime Database, a NoSQL cloud-based solution that keeps data synced across clients instantly. Later, Google introduced Cloud Firestore, a scalable, document based database built on Google Cloud infrastructure.

Firebase's main goal is to simplify app development by offering an all-in-one platform that includes database services, user authentication, hosting, and analytics making it especially appealing for mobile and web developers. [24]

Firebase is commonly used to power mobile and web apps that require real-time functionality, such as chat applications, social networks, and collaborative tools. Its ability to synchronize data instantly across Android, iOS, and web clients makes it an excellent choice for cross platform development. In gaming and IoT, Firebase excels in scenarios where real time interaction is essential like live scoreboards, multiplayer experiences, and connected devices. It's also a favorite among startups and developers building prototypes, offering a fast, infrastructure free way to launch functional apps quickly. [24] [25]

2.3.4 MongoDB

MongoDB was created in 2007 by 10gen, which later became MongoDB Inc. It's a document-based NoSQL database built to handle flexible, fast-changing data. Instead of traditional tables and rows, MongoDB stores information in JSON-like documents, making it ideal for working with unstructured or semi-structured data. [26]

***Figure 17**: MongoDB logo [70]*

MongoDB is widely used in real time platforms such as Uber and LinkedIn, where rapid access to constantly changing data is critical. It also plays a key role in sectors like healthcare and finance, enabling the analysis of complex and unstructured datasets. Additionally, its scalability and flexibility make it a popular choice for content management systems and IoT networks that require efficient handling of large and diverse data streams. [27]

2.3.5 Cassandra

Apache Cassandra was originally developed by Facebook in 2008 to handle massive inbox search volumes. A year later, it became an open-source project under the Apache Software Foundation. Cassandra is a wide-column NoSQL database designed for high availability, fault tolerance, and scalability, making it ideal for managing large volumes of data across distributed systems. [28]

***Figure 18**: Cassandra logo [61]*

Cassandra is trusted by major companies such as Instagram, Apple, and Netflix to manage large-scale, data-intensive workloads with reliability and speed. It's particularly well-suited for IoT platforms, log management systems, and time-series data application environments where performance, scalability, and continuous uptime are essential.

2.3.6 Redis

Figure 19: Redis logo [71]

Redis short for Remote Dictionary Server, was developed by Salvatore Sanfilippo in 2009. While it started out as a caching solution, it quickly grew into a powerful, general purpose in memory data store. Redis supports a variety of data structures, including strings, lists, sets, hashes, and even publish/subscribe messaging making it incredibly flexible for real-time applications. [29]

Redis is commonly used to manage user sessions in web applications, offering fast and reliable performance. Its speed and flexibility also make it ideal for powering real time leaderboards, analytics dashboards, and other time-sensitive features. Additionally, Redis often serves as a high-speed caching layer, significantly improving application responsiveness and reducing database load. [30]

2.3.7 MariaDB

Figure 20: MariaDB logo [62]

MariaDB was launched in 2009 by the original creators of MySQL, following Oracle's acquisition of MySQL. Designed as a fully open-source alternative, MariaDB remains compatible with MySQL while offering enhanced performance, improved storage engines, and continued community driven development. [31]

MariaDB is trusted by major organizations such as Deutsche Bank, Google Cloud, and Wikipedia for its reliability and performance. It's widely adopted in e-commerce platforms, financial systems, and serves as a seamless drop-in replacement for MySQL in existing projects, making it a practical choice for teams looking to maintain compatibility while benefiting from ongoing innovation. [32]

2.4 Application Programming Interfaces (APIs)

Application Programming Interfaces (APIs) play a crucial role in modern web development by allowing different systems and services to communicate seamlessly. They're especially valuable in platforms like event management systems, where real-time data exchange, secure authentication, analytics, and third-party integrations are essential.

APIs help keep systems modular and scalable, making it easier to adapt to new features or evolving user needs. In this context, they serve as the building blocks that connect core functionalities with external tools and services.

2.4.1 REST API

The REST API acts as the bridge between a platform's front end and its backend logic, enabling smooth communication through standard HTTP methods like GET, POST, PUT, and DELETE. It follows a straightforward request-response pattern, allowing clients to fetch or update resources while the server returns data typically in a widely supported format like JSON (as shown in Figure 26).

***Figure 21**: REST API logo [64]*

In Eventia, the REST API is at the heart of key operations, such as creating events and managing platform data. It ensures structured, consistent communication between different components of the system, making the platform more reliable and maintainable.

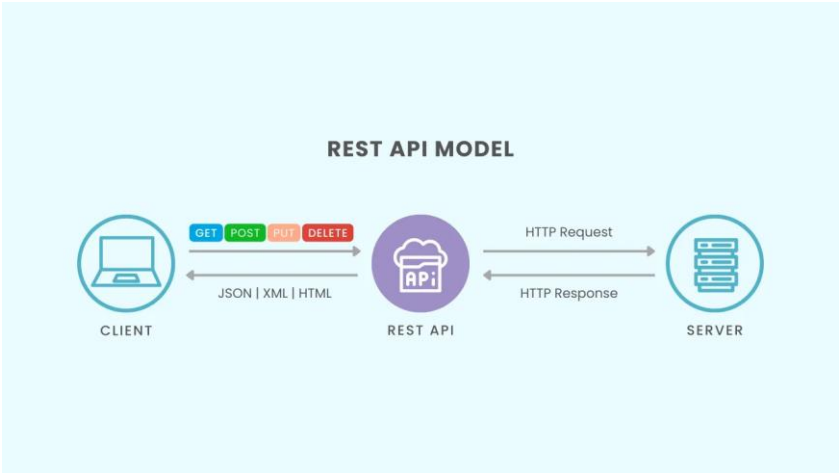

***Figure 22**: Control flow of REST API [63]*

2.4.2 Firebase Authentication API

The Firebase Authentication API offers a fully managed identity service that streamlines secure logins, user verification, and access control. It supports a variety of sign-in methods, including email and password, phone

***Figure 23**: Firebase Authentication API logo [65]*

number verification, and third party providers like Google making authentication both flexible and user-friendly. [33]

In Eventia, this API plays a key role in enforcing role-based access control by assigning custom claims to users. This allows different user roles such as administrators, organizers, vendors, and attendees to have tailored levels of access based on their responsibilities. By verifying ID tokens on the server before handling any requests, Firebase Authentication ensures that only authorized users can access sensitive parts of the system. This not only protects user data but also reinforces the platform's overall security.

2.4.3 Google Analytics API

***Figure 24**: Google Analytics API logo [66]*

The Google Analytics API allows platforms to collect, process, and visualize data about user behavior and system performance, offering valuable insights into how users interact with an application. It can track actions like registrations, ticket purchases, and event check ins, while also generating reports that reveal trends in user engagement and participation.

2.4.4 Google Maps API

***Figure 25**: Google Maps API logo [67]*

The Google Maps API brings powerful mapping and location-based services to the platform, making it easy to integrate geographic features directly into the user experience. It supports key functions like embedding interactive maps, providing step by step directions, and displaying detailed place information all of which help users navigate events more easily and interact with location-based content in a meaningful way.

2.4.5 WebSocket API

***Figure 26**: WebSocket API logo [68]*

Unlike REST APIs, which rely on a request–response model, this approach establishes a persistent, two way communication channel between the client and server (see Figure 27). This continuous connection allows data to flow instantly in both directions, making it ideal for features that demand real-time updates, such as live chat, notifications, or dynamic dashboards. [34]

***Figure 27**: Control flow of WebSocket API [69]*

Chapter 3: Literature Review

In this chapter, existing solutions within the domain of event management platforms are examined to provide a comprehensive understanding of the current landscape. Analyzing and comparing the strengths and weaknesses of these competitors not only highlights prevailing best practices but also uncovers significant gaps and limitations in the market.

3.1 Existing solutions in event management field

3.1.1 Eventbrite

Eventbrite is an event management platform that provides an easy way for organizers to create and customize event pages, manage registrations, and sell tickets. The platform supports different ticket types (free, paid, VIP, discounts) and integrates with secure payment systems.

***Figure 28**: Eventbrite logo [35]*

It also includes marketing tools, such as social media sharing, email invitations, and its own event discovery marketplace, which helps attract attendees. Eventbrite offers analytics and reports for tracking sales, revenue, and attendee demographics, along with onsite tools like mobile check-in and QR code scanning.

Eventbrite supports a wide range of event types, including concerts, classes and workshops, festivals and fairs, conferences, corporate events, and online events. It also caters to different industries, such as music, food and beverage, performing arts, charity and causes, and retail. This variety makes it

suitable for both entertainment-based events and professional or community focused gatherings.

The pricing model is free for free events, but for paid events, Eventbrite charges a service and processing fee per ticket. In addition, Eventbrite offers paid subscription plans (such as Professional and Premium) that provide access to more advanced features for larger or recurring events. Its strengths are its user friendly interface, global reach, and strong brand recognition. However, it can become expensive for large-scale events and lacks some advanced features needed for complex conferences.

Overall, Eventbrite is best for small to medium-sized events and is a strong reference point in ticketing and promotion features for event management systems. [35]

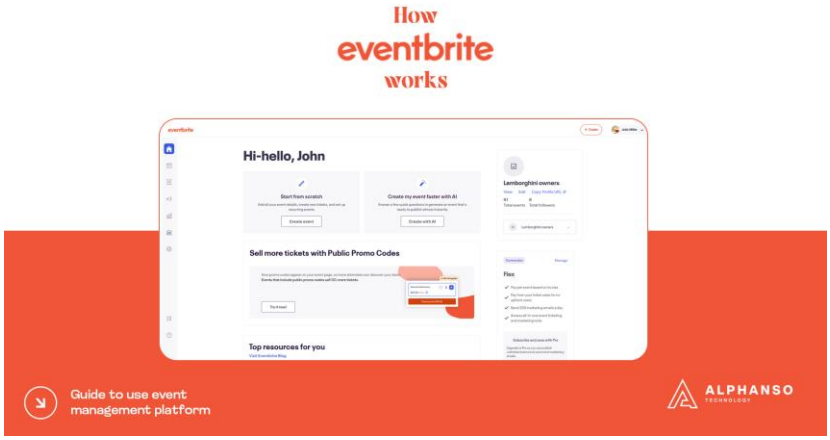

***Figure 29**: Eventbrite user interface [36]*

3.1.2 Cvent

***Figure 30**: Cvent logo [37]*

Cvent is an event management platform that supports in person, virtual, and hybrid events through a wide range of integrated tools. It enables organizers to create event websites, manage registrations, design agendas, and offer multi type tickets with built-in payment options.

The platform also provides advanced analytics dashboards for tracking registrations, attendance, and engagement, as well as onsite solutions such as mobile check in, badge printing, and seating management. Cvent is widely adopted by enterprises, government agencies, and institutions that require reliable and scalable event solutions.

Although Cvent is valued for its robust features and support for complex events, it is often considered expensive for smaller events and has a steep learning curve, making it best suited for medium to large scale or recurring events.

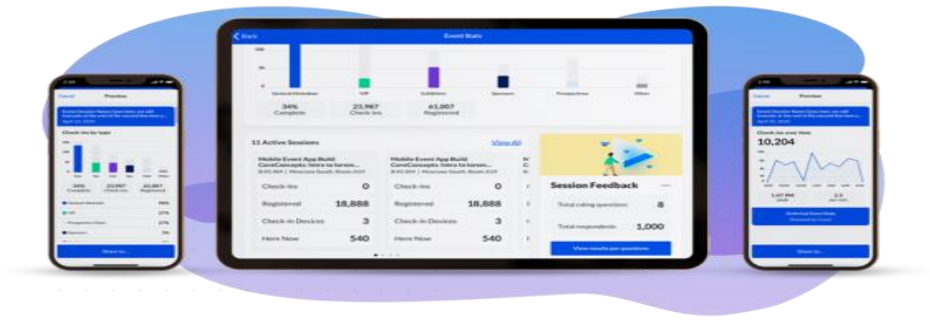

***Figure 31**: Cvent user interface [37]*

3.1.3 Whova

Whova is an integrated event management solution designed to enhance attendee engagement and streamline organizational processes. The platform provides an award-winning event app, online registration, event marketing, and a suite of management tools. Its primary goal is to improve networking opportunities, extend event experiences beyond physical boundaries, and reduce logistical inefficiencies. Event organizers benefit from Whova by mobilizing event information, reducing printing costs, and offering enhanced exposure to sponsors and exhibitors. Attendees use the app to browse event brochures, personalize schedules, and network before and after events, while sponsors gain increased visibility and lead generation opportunities.

***Figure 32**: Whova logo [45]*

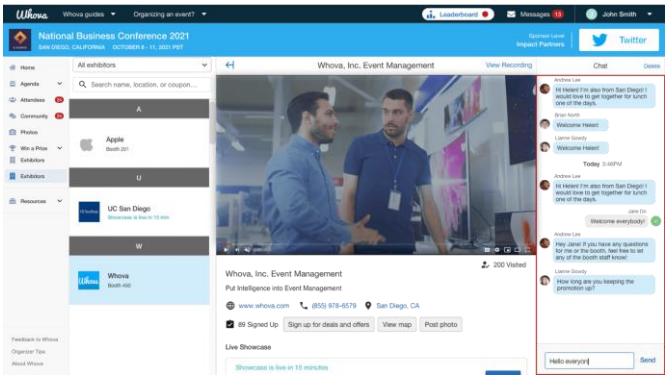

***Figure 33**: Whova user interface [38]*

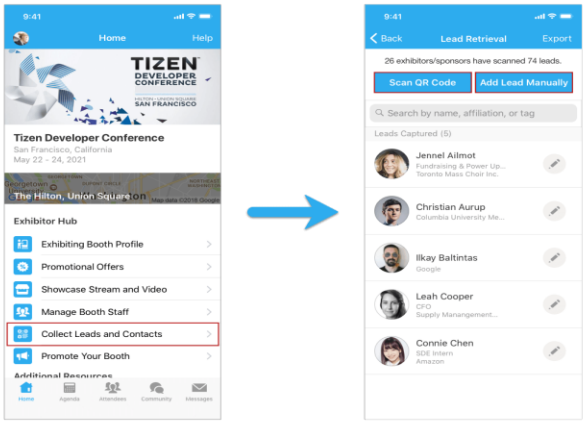

***Figure 34**: Whova mobile user interface [38]*

3.1.4 Bizzabo

Bizzabo is an event management software platform that supports immersive in-person, virtual, and hybrid experiences. Its Event Experience Operating System is a data rich, open platform that enables organizers to plan events, engage participants, and foster communities while safeguarding attendee data. Bizzabo powers events for world-leading brands, including Fortune 100 enterprises, financial institutions, creative agencies, and scaling technology companies, distinguishing itself through its focus on data-driven engagement and business outcomes. [39]

***Figure 35**: Bizzabo logo [47]*

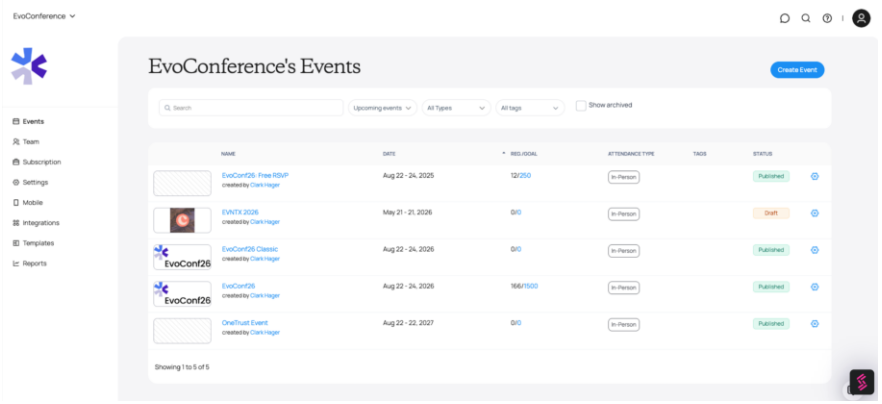

***Figure 36**: Bizzabo user interface [47]*

3.1.5 Accelevents

Accelevents is an enterprise-grade platform that provides end-to-end solutions for events of varying scales, whether in-person, virtual, or hybrid. It offers extensive customization of registration workflows, including conditional logic and multiple ticketing options.

The platform's onsite capabilities include expedited check in, self service kiosks, and badge printing. In addition, Accelevents integrates exhibitor and sponsor management, lead-capture tools, mobile event applications, and real-time analytics. Integration with customer relationship management (CRM) and marketing systems further enhances its functionality, while 24/7 customer support and white-label options provide flexibility and branding control for organizers. [40]

***Figure 37**: Accelevents logo [40]*

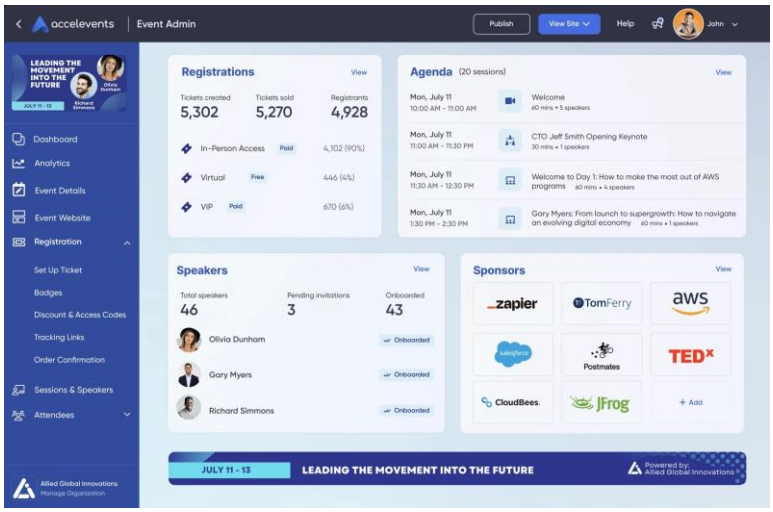

***Figure 38**: Accelevents user interface [41]*

3.1.6 RingCentral Events

RingCentral Events, formerly branded as Hopin, is a virtual event platform purpose-built for fully integrated online event experiences. It supports the complete event workflow, from registration and branding to live streaming, attendee engagement, analytics, and post-event insights. The platform includes customizable landing pages, interactive networking tools, real-time chat and Q&A, exhibitor booths, and integrated data dashboards. These features enable a seamless and engaging experience for both organizers and attendees. [42]

***Figure 39**: RingCentral Events logo [42]*

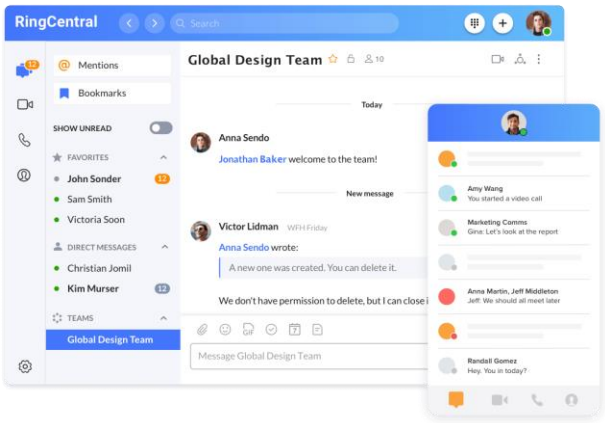

***Figure 40**: RingCentral Events user interface [42]*

3.1.7 EventXPro

EventXPro is a comprehensive event management platform that provides solutions for physical, virtual, hybrid events, and webinars. It supports the entire event lifecycle, including attendee registration, session scheduling, speaker coordination, real-time check-ins, badge

***Figure 41**: EventXPro logo [43]*

printing, and post-event analytics. The platform emphasizes scalability and user-friendliness, offering tailored features for diverse event formats and roles to enhance overall engagement. [43]

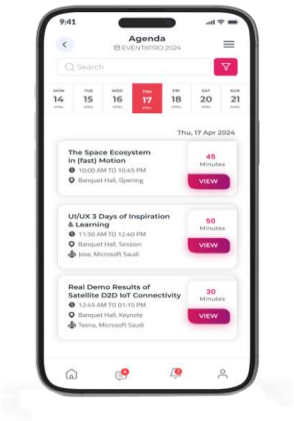

***Figure 42**: EventXPro user interface [43]*

3.1.8 Odoo Events

Odoo Events is part of the broader Odoo business application suite, designed to manage the organization, promotion, and delivery of events. The module enables event planners to create customizable event pages, manage speaker proposals and schedules, and handle ticketing with multiple tiers. It also provides tools for sponsorship management and exhibitor coordination. Integrated with Odoo's CRM, eCommerce, and marketing modules, Odoo Events ensures seamless workflows, while supporting analytics integration through Google Analytics to track attendee behavior and event performance. [44]

***Figure 43**: Odoo Events logo [44]*

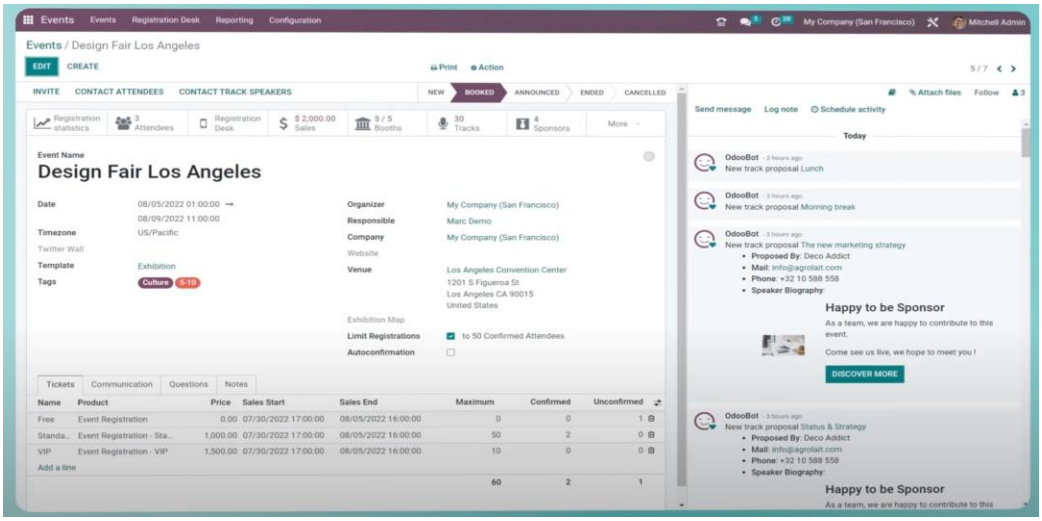

***Figure 44**: Odoo Events user interface [44]*

3.2 Comparison among event management platforms

Table 1: *Comparison among event management platforms*

Platform / Feature	Eventbrite	Cvent	Whova	Bizzabo	Accelevents	RingCentral	EventXPro	Odoo
Registration	Direct	Approval based	Direct	Direct	Approval based	Direct	Direct	Direct
Ticketing	Multi-type +VIP	Multi-type +VIP	Multi-type	Multi-type +VIP	Multi-type +VIP	Multi-type	Multi-type	Multi-type +VIP
Check-in	Scanning, Manual	Kiosk, QR Code, Manual	Scanning, Manual	Kiosk, Scanning, Manual	Via Platform , Kiosk, Scanning, Manual	No info.	Via Platform , Kisok, Scanning, Manual	No info.
Check-out support	No	Yes	No	Yes	Yes	Yes	Yes	Yes
Analytics	Reports	Dashboards	Reports	Dashboards	Real-time Dashboards	Dashboards	Reports + Dashboards	External Tools: Google Analytics + Odoo BI
Offline Mode	Limited	Yes	Limited	Yes	Yes	No	Yes	No
Supporting SCEGA Regulations	No	No	No	No	No	No	Limited	Limited
Vendor Collaboration	No invitation approach	Two-way	One-way	Two-way	Two-way	Two-way	Two-way	One-way
Arabic Support	No	No	No	No	No	No	No	Yes

3.3Limitations of Existing Event Management Platforms

While existing event management platforms offer a wide range of features, they present several critical shortcomings when evaluated against the requirements of the Saudi market. A major limitation is the lack of Arabic language support and localized interfaces, which makes many global platforms less accessible to organizers, vendors, and attendees within the Kingdom. Also, offline functionality is either limited, or absent which causing make users struggle when recheck the event details, reducing the reliability of these solutions in environments where internet connectivity is inconsistent.

Another challenge is the inconsistency in vendor collaboration mechanisms. Some platforms only allow organizers to invite vendors, while others provide basic exhibitor listings without structured approval workflows. This restricts the flexibility needed for managing diverse stakeholders such as sponsors, service providers, and logistics partners.

Most importantly, current platforms are not built to comply with the regulations and licensing requirements of the Saudi Conventions & Exhibitions General Authority (SCEGA). Compliance with these regulations is mandatory for hosting official events in the Kingdom, covering aspects such as event licensing, attendee registration records, exhibitor approvals, venue safety standards, and the inclusion of license numbers in promotional materials. The absence of such regulatory alignment creates a significant gap, as organizers risk delays, penalties, or rejection of their events if they rely on platforms that do not adhere to SCEGA's framework.

These limitations highlight the need for a new solution designed specifically for the Saudi market one that integrates regulatory compliance with SCEGA, supports Arabic interfaces, ensures offline reliability, and fosters more inclusive vendor collaboration. Such a system addresses current market gaps and empowers organizers to deliver events that are both compliant and efficient.

Chapter 4: System Analysis

4.1 Functional Requirements

[FR1.0] The application shall provide an interface for users to login or signup based on their role.

- [1.1] When opening the application, the user can browse the website freely, with login and signup options available in the website header for easy access.
  - [1.1.1] Log in or sign up
  - [1.1.2] If the user chose to sign up, the user is shown four role options: Organizer, Vendor, Attendee, or Guest.
  - [1.1.3] Log in options.
- [1.2] The system shall provide two ways for users to log in, either by registered account or guest account.
  - [1.2.1] If the user chooses the guest option or selects 'Continue as Guests", the system allows limited access without registration.
    - [1.2.1.1] Guests can browse the website, explore available events, and view event descriptions.
    - [1.2.1.2] Guests can register for events by providing a valid email or phone number to receive a digital ticket.
    - [1.2.1.3] Guests cannot submit feedback, or access personalized features.
  - [1.2.2] For registered accounts, the system requires users to log in using their email and password.
    - [1.2.2.1] The system validates the provided credentials.
    - [1.2.2.2] The system checks the user's assigned role (Organizer, Vendor, or Attendee).
    - [1.2.2.3] System redirects the user to the appropriate dashboard based on their role.
- [1.3] If the organizer option is chosen, the system proceeds based on the user's selection
  - [1.3.1] The organizer must provide full name, email, and password
  - [1.3.2] The system validates all fields before account creation.
  - [1.3.3] Organizer can reset their password through registered email and phone number.
- [1.4] If the vendor option is chosen, the system proceeds based on the user's selection
  - [1.4.1] The vendor must provide organization name, email, and password.
  - [1.4.2] The system validates all fields before account creation.
  - [1.4.3] Vendors can reset their password through email or phone number.
- [1.5] If the Attendee option is chosen, the system proceeds based on the user's selection
  - [1.5.1] The Attendee must provide username, full name, password and either email or phone number.
  - [1.5.2] The system validates that all mandatory fields are registered.

    - [1.5.3] Attendees can recover their password via registered email or phone number.
- [1.6] The system provides password recovery for registered users (Organizer, Vendor, Attendee)
    - [1.6.1] Users can request password reset via registered email or phone number.
    - [1.6.2] The system validates the recovery request before allowing a password change.

[FR2.0] The application shall provide organizers to perform event management operations through a dedicated interface.

- [2.1] The system shall allow organizers to create new events by entering essential details such as title, description, date, time, ticket price, and location.
    - [2.1.1] The system validates that all mandatory fields are completed before saving the event.
    - [2.1.2] Organizers can select the event category or type.
    - [2.1.3] Organizers can upload an event banner or image to represent the event.
    - [2.1.4] The system displays a confirmation message once the event is successfully created.
- [2.2] The system allows organizers to manage events after creation.
    - [2.2.1] Organizers can view a list of all their created events..
    - [2.2.2] Organizers can update event details such as title, date, time, description and location.
    - [2.2.3] The system ensures that all updates are reflected for attendees and vendors automatically.
    - [2.2.4] Organizers can delete events, and the system notifies registered attendees and vendors of the cancellation.
- [2.3] The system enables organizers to invite and manage vendors for each event.
    - [2.3.1] Organizers can send digital invitations to vendors directly from the system containing event details such as event title, date, and venue.
    - [2.3.2] Organizers can view the current status of all invitations (e.g., Pending, Accepted, Rejected).
    - [2.3.3] The system records the date and time of each vendor's response for reference.
    - [2.3.4] Organizers can resend invitations or withdraw them before response.
    - [2.3.5] Organizers can contact vendors through the integrated communication module.
- [2.4] The system shall provide organizers with tools to generate post-event analytical reports and collect attendee feedback survey.
    - [2.4.1] The system shall generate analytical reports after each event concludes, providing organizers with performance.
    - [2.4.2] The system shall allow organizers to view both newly generated and past event reports through a dedicated reports dashboard.
    - [2.4.3] The system shall allow organizers to create post-event feedback survey to gather attendee opinions, satisfaction and suggestions for improvement.

[FR3.0] The application shall provide attendees with features to explore, register for, and interact with events.

- [3.1] The attendee can browse a list of available events.
  - [3.1.1] The system displays key event details such as title, description, date, time, location, and ticket availability
  - [3.1.2] The attendee can search or filter events by name, type, or date.
- [3.2] The system shall allow attendees to register for available events through the event details page.
  - [3.2.1] The system validates the attendee's registration and ensures ticket availability.
  - [3.2.2] A digital ticket is generated upon successful registration.
    - [3.2.2.1] The ticket shall include essential information such as event name, date, time, location.
    - [3.2.2.2] The system shall store the ticket in the attendee's account for later access.
    - [3.2.2.3] The attendee shall receive a confirmation notification via email or in-app message.
  - [3.2.3] The system shall record the attendee's participation in their personal event history if the digital ticket is scanned.
  - [3.3.4] Once registration is completed, the attendee cannot cancel their ticket or withdraw from the event.
  - [3.3.5] The system shall prevent duplicate registrations for the same event by the same user.
- [3.3] The attendee can manage their profile information.
  - [3.3.1] The attendee can upload or update a profile picture (PFP).
  - [3.3.2] The attendee can edit personal details such as name, contact information, and password.
- [3.4] After attending an event, the attendee shall be allowed to rate the event, submit complaints and fill the event survey.
  - [3.4.1] The system shall provide the ability to rate the event on a five-star scale based on overall satisfaction,
    - [3.4.1.1] The system shall calculate the average ratings and display it.
  - [3.4.2] The system shall provide a default structured survey and link each response to the corresponding event record.
    - [3.4.2.1] The organizer shall be allowed to edit the survey questions.
- [3.5] The attendee can view a history of past events they have attended.
  - [3.5.1] Each record includes event name, date, location, and feedback submission status.

[FR4.0] The application shall provide vendor and stakeholder management functionalities.

- [4.1] The system shall allow vendors to respond to organizer invitations.
  - [4.1.1] Vendors shall be able to accept or reject invitations digitally through their dashboards.

    - [4.1.2] Upon acceptance, the vendor's status shall update to "Approved", and the organizer shall be notified automatically.
    - [4.1.3] If the invitation is rejected, the system shall record the response and notify the organizer with the reason provided (if any).
- [4.2] The system shall allow vendors to manage their active and past event participations.
    - [4.2.1] Vendors shall be able to view a list of all events they are currently participating in or have participated in previously.
    - [4.2.2] Vendors shall have the option to withdraw from upcoming events before the event start date.
    - [4.2.3] When a vendor withdraws, the system shall update the event status and notify the organizer automatically.
- [4.3] The system shall integrate vendor collaboration data within event analytics for better evaluation of event partnerships.
    - [4.3.1] Accepted vendors shall appear in event summaries and performance reports.
    - [4.3.2] Vendor interaction data shall be used to generate participation metrics in post-event analysis.

[FR5.0] The application shall support event communication and collaboration tools.

- [5.1] The system shall include an internal messaging feature that enables real-time communication between organizers, vendors, and attendees within the platform.
    - [5.1.1] The internal messaging refers to a built-in chat module integrated into the platform, allowing users to exchange messages, updates, and inquiries without using external email or third-party tools.
    - [5.1.2] The messaging module shall use WebSocket-based communication to ensure real-time message delivery and updates.
    - [5.1.3] Organizers shall be able to send announcements and private messages to vendors and attendees participating in their events.
    - [5.1.4] Vendors shall communicate with organizers to discuss logistical and service-related matters directly through the same module.
    - [5.1.5] Attendees shall be able to contact organizers through a "Contact Organizer" option available on the event details page.
    - [5.1.6] The system shall automatically route attendee inquiries to the appropriate organizer's dashboard and display them in a Support Chat section.
    - [5.1.7] Organizers shall receive instant notifications and alerts whenever new messages or inquiries are received.
- [5.2] The system shall store all communication records securely for reference and auditing.
    - [5.2.1] Each message shall include sender ID, recipient ID, timestamp, and message content.
    - [5.2.2] Communication data shall be encrypted in transit and at rest to ensure confidentiality.

    - [5.2.3] The system shall allow authorized users to retrieve previous messages for event documentation.
- [5.3] The system shall integrate communication logs into event analytics for performance and engagement insights.
    - [5.3.1] The system shall measure message frequency and response time as part of engagement metrics.
    - [5.3.2] Communication data shall be accessible to organizers through analytical dashboards.

[FR6.0] The application shall provide analytical reports to assist organizers in evaluating event outcomes.

- [6.1] The system generates automated analytical reports summarizing event performance after completion.
    - [6.1.1] Reports include attendance data, statistics about demographics and feedback summaries.
    - [6.1.2] The system visualizes insights through charts or dashboards that help organizers identify event strengths and weaknesses.
- [6.2] Organizers can download reports in multiple formats (PDF, Excel) for documentation and future planning.
- [6.3] The system shall allow organizers to access previous event analytics for trend comparison.

[FR7.0] The application shall integrate with the Saudi Conventions and Exhibitions General Authority (SCEGA) for authentication and compliance.

- [7.1] There shall be a verification process for organizers
    - [7.1.1] The system connects to SCEGA's authentication gateway to verify organizer identity before event creation or publication.
- [7.2] Organizers must provide their SCEGA license number or authorization credentials during event creation process.
- [7.3] There shall be an authorization control
    - [7.3.1] Only verified organizers can create and publish events.
    - [7.3.2] The system blocks event creation if verification fails and displays a clear notification.

4.2 Non-Functional Requirements

[NFR1.0] Compliance with Saudi Laws and Regulations
The system shall adhere to all applicable Saudi Arabian laws, regulations, and standards related to data protection, cybersecurity, and event management. This includes:

- [1.1] Saudi Data Protection Law (PDPL): All personal and sensitive data must be collected, processed, and stored in compliance with PDPL to ensure privacy and legal integrity.
- [1.2] Saudi Cybersecurity Framework (NCA ECC): The system shall implement controls mandated by the National Cybersecurity Authority to safeguard data confidentiality, integrity, and availability.
- [1.3] Saudi Electronic Transactions Law: All digital approvals, invitations, and transactions shall be legally recognized under this law.
- [1.4] Saudi Authority-Specific Regulations: The system shall support compliance with requirements from entities such as the Saudi Conventions & Exhibitions General Authority (SCEGA) and other relevant government bodies.

[NFR2.0] Platform Compatibility and Responsiveness
The system shall be accessible and fully functional across multiple devices and platforms, ensuring a consistent user experience. This includes:

- [2.1] Responsive Design: The interface shall adapt seamlessly to various screen sizes, including desktops, tablets, and smartphones.
- [2.2] Cross-Browser Compatibility: The system shall perform consistently across major browsers such as Chrome, Firefox, Safari, and Edge.

[NFR3.0] Security and Data Protection
The system shall implement robust security measures to protect user data, transaction details, and event-related information. This includes:

- [3.1] Data Encryption: All sensitive data in transit and at rest shall be encrypted using TLS and AES-256 standards.
- [3.2] Authentication and Authorization: Multi-factor authentication (MFA) and role-based access control (RBAC) shall be enforced to restrict system access based on user roles.
- [3.3] Audit Logs: The system shall maintain detailed logs of user activities for security auditing and compliance reporting.
- [3.4] Secure APIs: All integrated APIs and third-party services shall adhere to strict security protocols to prevent unauthorized access.

[NFR4.0] Usability and User-Centered Design
The system shall be intuitive and easy to use for all stakeholders, including organizers, vendors, sponsors, and attendees. This includes:

- [4.1] Intuitive Navigation: The user interface shall be clean, logically organized, and require minimal training to use.
- [4.2] Role-Based Dashboards: Each user role shall have a customized dashboard with relevant tools and information.
- [4.3] Accessibility: The system shall adhere to WCAG guidelines to support users with disabilities.

- [4.4] User Error Handling: The system shall provide clear, informative, and user-friendly error messages to guide users in case of failures, such as invalid input or connection issues.

[NFR5.0] Arabic Language and Localization
The system shall fully support Arabic language and cultural conventions to ensure accessibility and relevance within Saudi Arabia. This includes:

- [5.1] Right-to-Left (RTL) Layout: The interface shall support RTL text orientation for Arabic content.
- [5.2] Arabic Date and Number Formats: Dates, numbers, and units shall be displayed according to local conventions.
- [5.3] Cultural Relevance: All content and design elements shall respect local cultural norms and practices.

[NFR6.0] Performance and Scalability
The system shall perform efficiently under expected user loads and be scalable to accommodate growing numbers of events and users. This includes:

- [6.1] Load Handling: The system shall support concurrent access by multiple stakeholders during high-demand periods such as event registration or ticket sales.
- [6.2] Response Time: Key operations (e.g., ticket booking, dashboard loading) shall have a response time of under 3 seconds.
- [6.3] Scalable Architecture: The system shall be designed to scale horizontally to handle increases in data and user traffic.

Chapter 5: System Design

This chapter presents the overall design of the proposed system, translating requirements into a clear technical framework. It begins with the use case diagrams and detailed use case specifications, which define the system's functional boundaries and describe how users interact with each feature. The activity diagrams illustrate the internal workflow and behavior of key processes. The system architecture section outlines the major components and how they communicate to deliver the required functionalities. The database design details the data models, entities, and relationships that support system operations. Finally, the UI prototype showcases the intended user interface layout and interaction flow, offering a visual representation of the system's final look and feel.

5.1 Use-Cases

This section presents the system's use case diagrams, which describe how different users interact with the system to achieve their goals. Each use case represents a specific functionality that the system provides from the user's perspective. These diagrams help visualize the functional requirements, define the system boundaries, and clarify the roles of various actors involved. They also serve as a foundation for identifying key interactions and guiding subsequent stages of design, such as class and sequence diagrams.

Table 2 *A description of the main system roles and their functions*

Terms	Definitions
Attendee	A user who browses events, registers, receives digital tickets, and provides post-event feedback such as ratings or survey responses.
Guest	The guest who is able to browse and check events details but unable to register in events.
Organizer	The authorized user responsible for creating, managing, and publishing events and monitoring participant activities.
Vendor	The vendor refers to any external party or organization that provides services, resources, or logistical support necessary for an event's success.
System	The software solution that serves as the central coordinating environment where all stakeholders (organizers, vendors, sponsors, attendees, and authorities like SCEGA Gateway) interact digitally to perform their respective roles and get their needs.

SCEGA Gateway	An external verification service integrated with Eventia to authenticate organizers and validate event compliance with SCEGA regulations.

5.1.1 General Use-case diagram

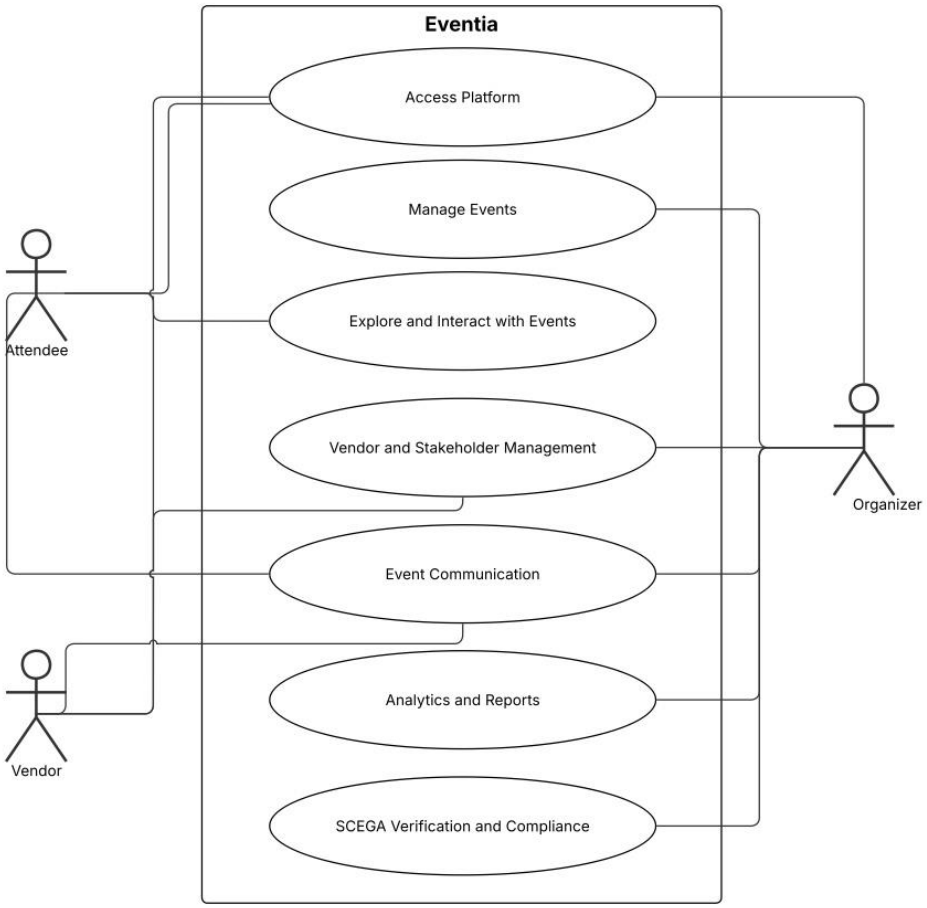

Figure 45 *General use-case diagram for the system*

5.1.2 Access Platform

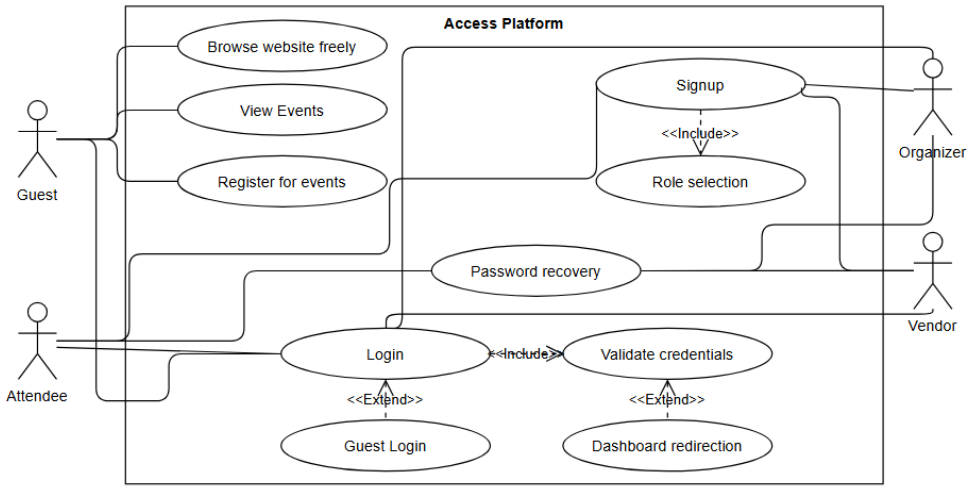

Figure 46 *Access Platform Use-case diagram*

5.1.3 Manage Events

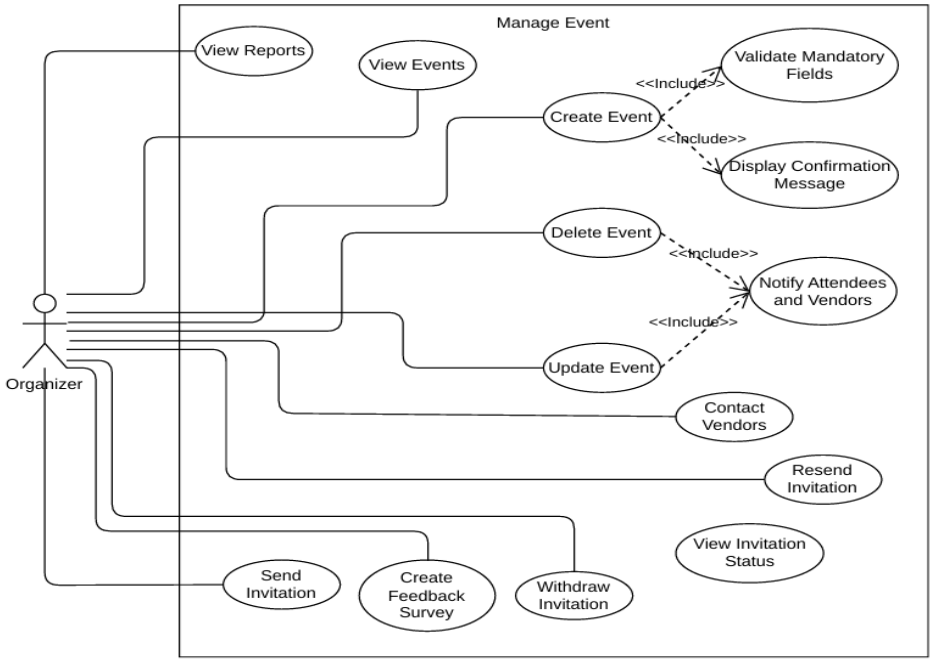

Figure 47 *Manage events use-case diagram*

5.1.4 Attendee Services Use-case diagram

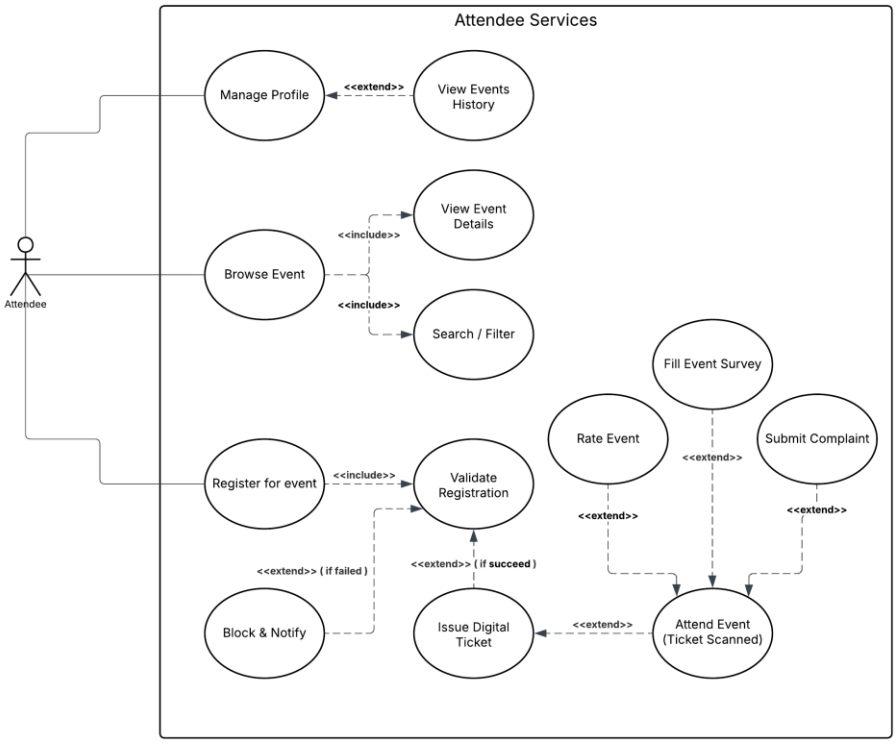

***Figure 48**. Attendee Services use-case diagram*

5.1.5 Vendor and Stakeholder Management

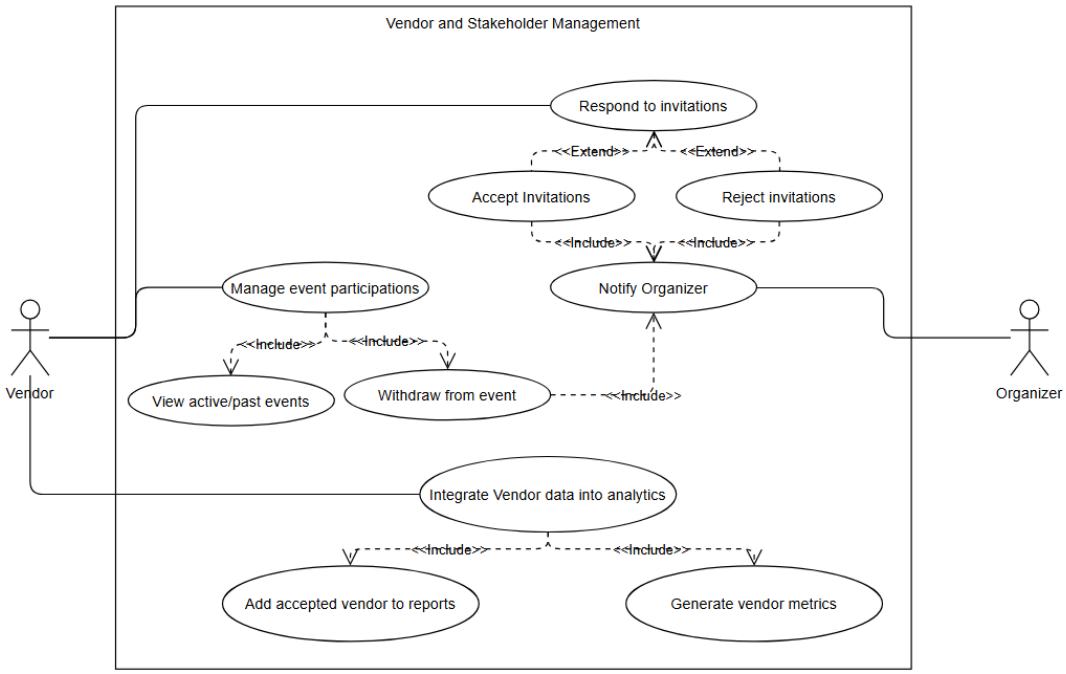

***Figure 49** Vendor and Stakeholders Management use-case diagram*

5.1.6 Analytical Reports

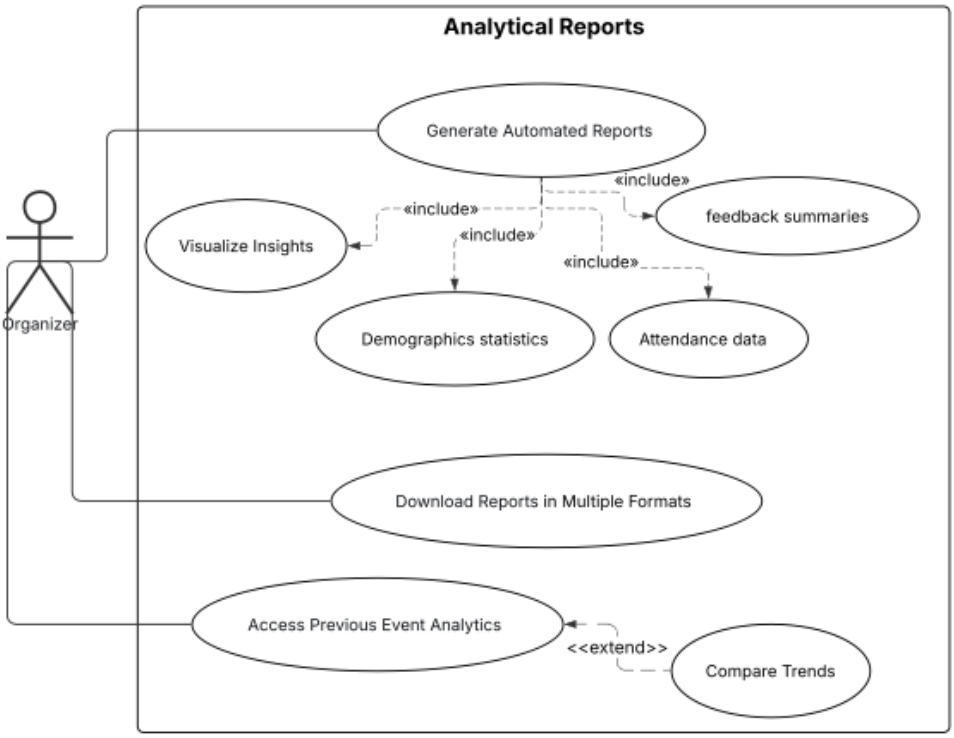

Figure 50 *Analytical Reports use-case diagram*

5.1.7 Communication and Collaboration

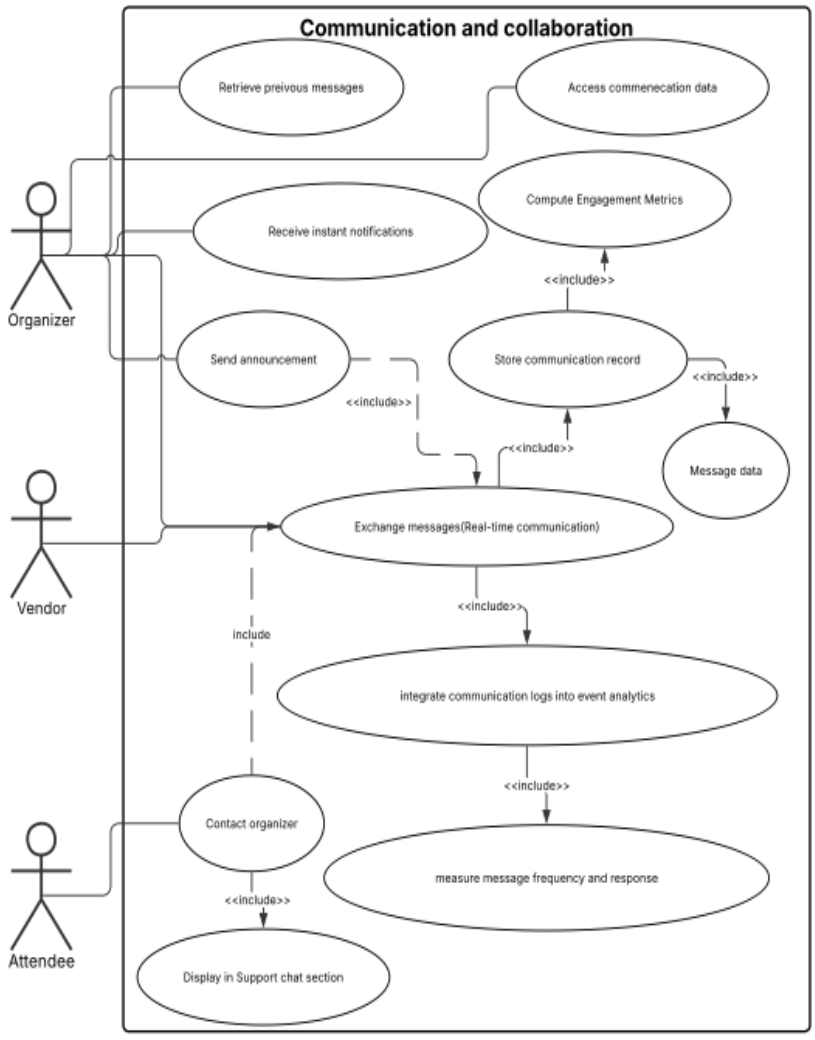

Figure 51 *Communication and collaboration use-case diagram*

5.1.8 SCEGA authentication & Compliance

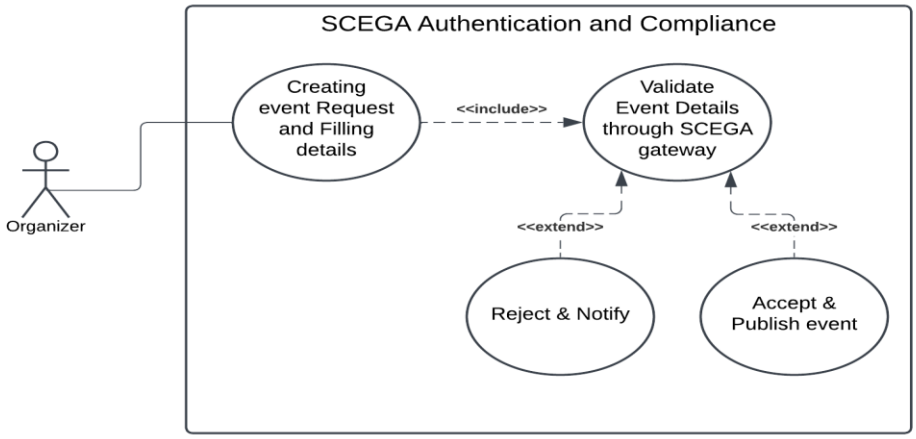

Figure 52 *SCEGA authentication & Compliance use-case diagram*

5.2 Use case Specification

This section provides detailed specifications for each use case identified in the system. Each specification describes the purpose, actors involved, preconditions, main flow of events, alternative or exceptional flows, and postconditions. These descriptions complement the use case diagrams by elaborating on the system's functional behavior and interactions.

Table 3 *Login to System use-case specification*

Use Case ID:	1		
Use Case Name:	Login to System		
Created By:	Abdullah Binradyan	**Last Updated By:**	
Date Created:	16-10-2025	**Date Last Updated:**	
Actors:	• User		
Description:	This use case describes how a registered user logs into the Eventia platform by providing valid credentials. The system verifies the login information and grants access to the corresponding dashboard based on the user's role.		
Trigger:	The user selects "Login" from the Eventia homepage or navigation menu.		
Preconditions:	The user must already have a registered account in the system. The user must have a valid email and password.		
Postconditions:	The system authenticates the user successfully. The user is redirected to their respective dashboard (Organizer, Vendor, or Attendee). A session is established to maintain login state.		
Normal Flow:	1. The user opens the Eventia application. 2. The system displays the Login page. 3. The user enters their email and password. 4. The user clicks the "Login" button. 5. The system validates the credentials against the database. 6. If the credentials are correct, the system retrieves the user's role. 7. The system redirects the user to the appropriate dashboard: • Organizer Dashboard • Vendor Dashboard • Attendee Dashboard 8. The system displays a personalized welcome message.		
Exception	1. Invalid Email or Password: The system displays an error message: "Incorrect email or password. Please try again."		

	2. Unregistered Email: The system shows: "No account found with this email. Please sign up first." 3. Account Locked or Disabled: The system displays: "Your account has been suspended. Please contact support." 4. Network or Server Error: The system displays: "Unable to connect. Please try again later."
Alternative Flows:	1. Forgot Password: If the user selects "Forgot Password", the system redirects them to the password recovery process (see Use Case: Recover a Password). 2. Remember Me: If the user checks "Remember Me", the system securely stores login credentials to maintain session persistence for future visits.
Includes:	[1.2.2]

Table 4 *Signe Up to the system use-case specification*

Use Case ID:	2		
Use Case Name:	Sign Up to the System		
Created By:	Abdullah Binradyan	**Last Updated By:**	
Date Created:	16-10-2025	**Date Last Updated:**	
Actors:	• User		
Description:	This use case describes how a new user creates an account on the Eventia platform. The user selects their role (Organizer, Vendor, or Attendee), provides required registration details, and submits the information. The system validates the input and creates a new user account with role-based permissions.		
Trigger:	The user selects "Sign Up" from the Eventia homepage or login screen.		
Preconditions:	The user is not currently registered in the system. The system is operational and connected to the user database.		
Postconditions:	A new user account is created successfully. The system stores the user's data securely in the database. The user receives a confirmation message and can now log in to the system.		
Normal Flow:	1. The user opens the Eventia web application. 2. The system displays the Home Page with "Sign Up" and "Login" options. 3. The user clicks "Sign Up." 4. The system displays a registration form and prompts the user to select a role:		

	• Organizer • Vendor • Attendee 5. The user selects their role. 6. The system displays the corresponding registration fields based on the selected role: • Organizer: Full name, email, password • Vendor: Organization name, email, password • Attendee: Username, full name, password, and email/phone number 7. The user fills in the required fields and clicks "Create Account." 8. The system validates the provided data: • Checks that all required fields are filled • Verifies the email format and password strength • Ensures the email is not already registered 9. If all validations pass, the system creates the new account and saves user details in the database. 10. The system displays a confirmation message: "Your account has been successfully created." 11. The user is redirected to the Login page or automatically logged in.
Exception	1. Missing Required Fields: The system highlights missing inputs and requests completion. 2. Invalid Email Format: The system displays an error: "Please enter a valid email address." 3. Weak Password: The system displays an error: "Password must meet the required complexity." 4. Duplicate Email: The system displays: "This email is already registered. Please log in or use another email." 5. Server or Network Error: The system shows: "Unable to process registration. Please try again later."
Alternative Flows:	1. Role Selection Change: If the user changes their role before submitting, the form resets to display the fields required for the new role. 2. Auto Login After Registration: Instead of redirecting to the login screen, the system automatically logs in the user after successful registration and opens their role-specific dashboard. 3. Email Verification (Optional): If email verification is enabled, the system sends a verification link to the user's registered email. The user must click the link to activate their account before logging in.
Includes:	[1.1.2], [1.3], [1.4], [1.5]

Table 5 *Recover a Password use-case specification*

Use Case ID:	3		
Use Case Name:	Recover a Password		
Created By:	Abdullah Binradyan	**Last Updated By:**	
Date Created:	16-10-2025	**Date Last Updated:**	
Actors:	• User		
Description:	This use case describes how a registered user who has forgotten their password can securely reset it using their registered email or phone number. The system verifies the provided information and allows the user to create a new password.		
Trigger:	The user selects “Forgot Password” from the login page.		
Preconditions:	The user must already have a registered account. The user must have a valid registered email or phone number stored in the system. The system’s email or SMS service must be available.		
Postconditions:	The user successfully resets their password. The new password is stored securely in the system. The user can log in with the new password.		
Normal Flow:	1. The user opens the Login page and clicks “Forgot Password.” 2. The system displays the Password Recovery page and prompts the user to enter their registered email address or phone number. 3. The user enters their information and submits the form. 4. The system verifies that the email or phone number matches a registered account. 5. If valid: • For Email: The system sends a password reset link. • For Phone: The system sends a verification code via SMS. 6. The user follows the provided instructions: • Email: Clicks the reset link to open the password reset page. • Phone: Enters the verification code into the system. 7. The system displays a form prompting the user to enter a new password and confirm it. 8. The user enters and submits the new password. 9. The system validates the new password against complexity requirements (e.g., length, characters, strength). 10. If valid, the system updates the password in the database and displays a success message: “Your password has been successfully reset.”		

	11. The user can now return to the login page and sign in with the new password.
Exception	1. Unregistered Email or Phone Number: The system displays: "No account found with the provided information." 2. Invalid Verification Code: The system displays: "Invalid code. Please try again or request a new one." 3. Expired Link or Code: The system shows: "Your reset link or code has expired. Please request a new one." 4. Weak Password: The system displays: "Password must meet the required complexity rules." 5. Server or Connection Error: The system displays: "Unable to process your request. Please try again later."
Alternative Flows:	1. Resend Verification Code or Link: If the user does not receive the email or SMS, they can click "Resend Code/Link." The system sends a new one and invalidates the previous request. 2. Cancel Password Recovery: The user can cancel the recovery process and return to the Login page at any time. 3. Admin Reset Assistance: If the user repeatedly fails to reset their password, they may contact the system administrator to manually reset it.
Includes:	[1.6.1], [1.6.2]

Table 6 *Continue as guest use-case specification*

Use Case ID:	4		
Use Case Name:	Continue as Guest		
Created By:	Abdullah Binradyan	**Last Updated By:**	
Date Created:	16-10-2025	**Date Last Updated:**	
Actors:	• Guest User		
Description:	This use case describes how a guest (unregistered user) can access the Eventia platform without creating an account. The system allows the guest to browse available events, view event details, and optionally register for an event by providing a valid email or phone number to receive a digital ticket.		
Trigger:	The user selects "Continue as Guest" from the homepage or login screen.		

Preconditions:	The user is not logged in or registered in the system. The Eventia application is available and connected to the event database.
Postconditions:	The guest is granted temporary access to the system. The guest can browse and view event details. If the guest registers for an event, their temporary contact information is stored for ticket delivery.
Normal Flow:	1. The user opens the Eventia web application. 2. The system displays the homepage with options: "Login," "Sign Up," and "Continue as Guest." 3. The user clicks "Continue as Guest." 4. The system grants limited access and redirects the guest to the Events page. 5. The guest can: • Browse all available events. • View event details such as title, description, date, location, and ticket availability. 6. The guest selects an event to view its details page. 7. If the guest chooses to register for the event, the system prompts them to provide a valid email address or phone number. 8. The guest enters their contact information and confirms. 9. The system validates the provided contact details. 10. The system generates a digital ticket and sends it to the guest's email or phone. 11. The system displays a confirmation message:"Your registration has been completed successfully. Your ticket has been sent."
Exception	1. Invalid or Missing Email/Phone Number: The system displays: "Please enter a valid email address or phone number." 2. Event Fully Booked: The system shows: "This event is no longer available for registration." 3. Server or Network Error: The system displays: "Unable to process your request. Please try again later." 4. Ticket Sending Failure: The system displays: "We couldn't send your ticket. Please check your contact details or try again."
Alternative Flows:	1. Guest Decides to Create an Account: At any time, the guest can click "Sign Up" from the top menu. The system transfers the guest to the registration form (see Use Case: Sign Up to the System). 2. Guest Browses Without Registering: The guest may continue exploring events without booking or entering any personal information. 3. Session Timeout: If the guest remains inactive for a specific period (e.g., 15 minutes), the system automatically ends the session and redirects them to the homepage.

Includes:	[1.2.1.1], [1.2.1.2]

Table 7 *Create New Event use-case specification*

Use Case ID:	5		
Use Case Name:	Create New Event		
Created By:	Turki AlMuayli	**Last Updated By:**	
Date Created:	15-10-2025	**Date Last Updated:**	
Actors:	• Organizer		
Description:	This use case allows an organizer to create a new event by entering essential details such as title, description, date, time, ticket price, and location. The system ensures that all required fields are filled before saving and displays a confirmation message after successful creation.		
Trigger:	The organizer selects “Create Event” from the event management interface.		
Preconditions:	The user must be registered in the system. The user must have a valid login (email and password).		
Postconditions:	Organizer must be logged into the system.		
Normal Flow:	1. Organizer navigates to “Create Event.” 2. The system displays a form to input event details. 3. Organizer enters title, description, date, time, ticket price, and location. 4. Organizer uploads an event banner or image. 5. Organizer selects the event category or type. 6. System validates all mandatory fields. 7. System saves the event to the database. 8. System displays a confirmation message indicating success.		
Exception	5. Required fields are missing : system displays an error message. 6. Invalid date/time format : system requests correction.		
Alternative Flows:	4. Organizer cancels event creation before submission.		
Includes:	[2.1]		

Table 8 *Manage Events use-case specification*

Use Case ID:	6		
Use Case Name:	Manage Events		
Created By:	Turki AlMuayli	**Last Updated By:**	

Date Created:	15-10-2025	**Date Last Updated:**	
Actors:	• Organizer		
Description:	This use case allows organizers to view, edit, update, or delete existing events. Changes made are reflected for both attendees and vendors automatically.		
Trigger:	when the organzier clicks on the “Manage Events”		
Preconditions:	Organizer must be logged in.		
Postconditions:	Event details are updated, deleted, or viewed successfully.		
Normal Flow:	1. Organizer opens the “Manage Events” section. 2. System displays a list of all created events. 3. Organizer selects an event to view or modify. 4. Organizer updates event details (title, date, time, description, location). 5. System validates and saves changes. 6. System notifies attendees and vendors of any updates or cancellations.		
Exception	1. No events found: system displays “No events available.” 2. Invalid update: system displays an error.		
Alternative Flows:	1. Organizer deletes an event: system confirms and notifies all participants.		
Includes:	[2.2]		

Table 9 *Invite and Manage Vendors*

Use Case ID:	7		
Use Case Name:	Invite and Manage Vendors		
Created By:	Turki AlMuayli	**Last Updated By:**	
Date Created:	15-10-2025	**Date Last Updated:**	
Actors:	• Organizer • Vendor		
Description:	This use case allows organizers to invite vendors to participate in events and manage their responses. Vendors can accept or reject invitations, and organizers can track invitation statuses.		
Trigger:	Organizer selects the “Invite Vendors” option for a specific event.		
Preconditions:	Organizer must be logged in.		

Postconditions:	Vendors receive invitations, and the responses are recorded and displayed to organizers.
Normal Flow:	1. Organizer selects an event and opens the “Invite Vendors” feature. 2. Organizer sends digital invitations containing event details. 3. Vendor receives and reviews the invitation. 4. Vendor accepts or rejects the invitation. 5. System updates the invitation status (Pending, Accepted, Rejected). 6. System records the date and time of each vendor’s response. 7. Organizer can resend or withdraw invitations if needed.
Exception	1. Vendor email invalid: system notifies the organizer. 2. Vendor does not respond: status remains “Pending.”
Alternative Flows:	1. Organizer contacts vendor via integrated chat module.
Includes:	[2.3], [4.1], [5.1]

Table 10 *Post-Event Reports and Feedback Surveys use-case specification*

Use Case ID:	8		
Use Case Name:	Post-Event Reports and Feedback Surveys		
Created By:	Turki AlMuayli	**Last Updated By:**	
Date Created:	15/10/2025	**Date Last Updated:**	
Actors:	• Organizer • Attendee		
Description:	This use case enables organizers to generate analytical reports and create post-event surveys to collect feedback from attendees. Attendees can submit their responses.		
Trigger:	The organizer selects “Reports & Feedback” for a completed event.		
Preconditions:	Event must be marked as “Completed.”		
Postconditions:	Reports are generated and stored. Surveys are created and linked to the event.		
Normal Flow:	1. Organizer accesses the “Reports & Feedback” section. 2. The system lists all completed events. 3. Organizer selects an event to generate a report. 4. The system compiles attendance, registration, and vendor data. 5. Organizer views the generated analytical report. 6. Organizer creates a feedback or complaint survey. 7. System links the survey to the event and publishes it for attendees. 8. Attendees receive notifications about the survey. 9. Attendees complete and submit their responses.		

	10. The system stores the responses and includes them in the analytical report.
Exception	1. Event not completed : report cannot be generated. 2. Missing data : system displays an error.
Alternative Flows:	
Includes:	[2.4], [3.4.2.1], [6.0]

Table 11 *Vendor and Stakeholder Management use-case specification*

Use Case ID:	9		
Use Case Name:	Vendor and Stakeholder Management		
Created By:	Abdullah Binradyan	**Last Updated By:**	
Date Created:	16-10-2025	**Date Last Updated:**	
Actors:	• Organizer • Vendor		
Description:	This use case describes how event organizers manage and collaborate with vendors and stakeholders through the Eventia platform. It includes sending invitations, tracking responses, approving or rejecting participation, and updating event records accordingly. Vendors can view invitations, respond digitally, and manage their active and past participations.		
Trigger:	The organizer sends or manages invitations for vendors through the event dashboard. The vendor receives a notification to respond to an invitation.		
Preconditions:	The organizer must be logged in and verified by the SCEGA system. The event must exist in the system. The vendor must have a registered and active account.		
Postconditions:	Vendors' responses (Accepted / Rejected / Withdrawn) are stored in the system. The event's vendor list and analytics are automatically updated. Notifications are sent to both organizer and vendor regarding any status changes.		
Normal Flow:	Organizer Actions: 1. The organizer logs into the system and opens the Event Dashboard. 2. The system displays a list of existing events. 3. The organizer selects an event and chooses "Manage Vendors." 4. The system displays the Vendor Management Panel with options to invite or track vendors. 5. The organizer clicks "Send Invitation."		

	6. The system shows a list of available vendors with details (name, type, service). 7. The organizer selects vendors and enters event details (title, date, venue, description). 8. The organizer clicks "Send." 9. The system sends digital invitations to the selected vendors. 10. The system records each invitation's status as Pending. Vendor Actions: 11. The vendor logs into their dashboard. 12. The system displays all received invitations. 13. The vendor opens an invitation to view event details. 14. The vendor selects "Accept" or "Reject." 15. The system records the vendor's decision and updates the status accordingly. 16. The organizer receives a notification of the vendor's response. 17. If Accepted, the vendor is added to the event's participant list. 18. If Rejected, the organizer can choose to invite an alternative vendor. Post-Event: 19. Vendors can view their past event participations. 20. Organizers can view vendor performance data in the event analytics report.
Exception	1. Vendor Account Inactive: The system displays: "Unable to send invitation. Vendor account is inactive." 2. Invalid Event Data: The system prevents sending invitations and displays: "Please verify event details." 3. Vendor Does Not Respond: After a set time (e.g., 48 hours), the system marks the status as No Response and notifies the organizer. 4. Organizer Cancels Event: The system automatically notifies all vendors that the event has been canceled. 5. System or Network Error: The system displays: "Unable to process your request. Please try again later."
Alternative Flows:	1. Resend or Withdraw Invitation: The organizer can resend an invitation or withdraw it before the vendor responds. The system updates the status accordingly and notifies the vendor. 2. Vendor Withdraws Participation: A vendor can withdraw from an accepted event before the start date. The system updates the participation status and notifies the organizer automatically. 3. Communication Between Organizer and Vendor: Either party can initiate chat through the integrated messaging feature for clarifications or updates.

	4. Auto Integration with Analytics: Upon event completion, the system automatically includes vendor collaboration data in the post-event performance report.
Includes:	[4.1], [4.2], [4.3]

Table 12 *Exchange Messages use-case specification*

Use Case ID:	10		
Use Case Name:	Exchange Messages (Real-time communication)		
Created By:	Abdulrahman Alghanmi	**Last Updated By:**	Abdulrahman Alghanmi
Date Created:	15/10/2025	**Date Last Updated:**	19/10/2025
Actors:	• Organizer • Vendor • Attendee		
Description:	This use case describes the process of a user composing and sending a single message to one or more recipients within the platform, enabling real-time, structured communication.		
Trigger:	User clicks the “Send” button to transmit a message.		
Preconditions:	1. The user must be registered and logged into the system. 2. The user must have the appropriate permissions to message the intended recipient.		
Postconditions:	1. The message is delivered to the recipient in real-time. 2. The message is stored securely in the database with a "Delivered" status. 3. The conversation thread is updated for all parties.		
Normal Flow:	1. The user opens the communication module or an existing chat. 2. The user composes a new message in the input field. 3. The user clicks the "Send" button. 4. The system validates the message (e.g., ensures it is not empty and is under the character limit). 5. The system assigns a unique ID to the message and tags it with metadata: Sender ID, Recipient ID(s), Timestamp, and Event ID. 6. Using WebSocket connections, the system immediately pushes the message to the recipient's interface. 7. The system stores the message in the encrypted database. 8. The system confirms the send was successful by displaying the message in the sender's chat window with a "Delivered" status		
Exception	1. Message is empty or too long: The system prevents sending and displays a warning: "Message cannot be empty" or "Message exceeds the maximum length."		

	2. Real-time connection lost**:** The system queues the message and automatically retries sending once the connection is restored. The user may see a "Connecting..." indicator.
Alternative Flows:	**1-** Edit a Message (Within a Short Timeframe): a. Immediately after sending, the user realizes a typo and clicks an "Edit" option on the sent message. b. The system allows the user to modify the message text within a short time. c. The user saves the changes. d. The system updates the message for both sender and recipient, and marks it as "Edited." 2- Reply to a Specific Message: a. The user hovers over a specific message in the chat history and selects a "Reply" option. b. The system quotes the original message in the new message input field. c. The user types their response and sends it. d. The system displays the new message as a threaded reply, providing context for the conversation
Includes:	[5.1.1], [5.1.2],[5.1.4],[5.1.5] [5.2.1], [5.2.2]

Table 13 *Generate Automated Reports use-case specification*

Use Case ID:	11		
Use Case Name:	Generate Automated Reports		
Created By:	Abdulrahman Alghanmi	**Last Updated By:**	
Date Created:	15/10/2025	**Date Last Updated:**	
Actors:	Organizer		
Description:	This use case is initiated automatically by the system after an event is completed. It compiles data from various sources into a comprehensive analytical report, visualizing key performance indicators for the organizer.		
Trigger:	The system's scheduler detects that an event's end date and time have passed, and its status is set to "Completed."		
Preconditions:	1. The event must be in the "Completed" state. 2. The event must have associated data (e.g., registrations, attendance scans).		
Postconditions:	1. A new analytical report is generated and saved in the database, linked to the event. 2. The report becomes available for the organizer to view in their analytics dashboard.		

Normal Flow:	1. The system's background service detects that an event has been completed. 2. The system automatically collects data from different modules related to the event: • Attendance Data • Demographic Statistics • Feedback Summaries 3. The system processes this data and generates visualizations (e.g., pie charts for demographics, bar graphs for attendance over time). 4. The system compiles these visualizations and data summaries into a structured report. 5. The system saves the report and links it to the event record. 6. The system notifies the organizer: "Your analytical report for [Event Name] is now ready."
Exception	1. Event has no data**:** The system will print message "No data available." 2. Data processing error: The system logs the error and retries the report generation after a short delay. If it fails again, it alerts system administrators.
Alternative Flows:	On-Demand Generation**:** An organizer can manually trigger this process for a completed event by clicking "Generate Report Now," which follows the same normal flow.
Includes:	[6.1]

Table 14 *Download Reports in Multiple Formats use-case specification*

Use Case ID:	12		
Use Case Name:	Download Reports in Multiple Formats		
Created By:	Abdulrahman Alghanmi	**Last Updated By:**	
Date Created:	15/10/2025	**Date Last Updated:**	
Actors:	Organizer		
Description:	This use case allows an organizer to export and save a generated analytical report to their local device in a chosen file format.		
Trigger:	The organizer clicks on the "Download" option while viewing a generated analytical report.		
Preconditions:	1. An analytical report for the event must have been generated 2. The organizer must be logged in and have view access to the event.		
Postconditions:	The selected report is downloaded to the organizer's local device in the chosen format.		

Normal Flow:	1. The organizer navigates to the Analytics dashboard and selects a completed event to view its report. 2. The organizer clicks the "Download" button. 3. The system displays a menu of available formats. 4. The organizer selects their desired format . 5. The system fetches the report data and visualizations. 6. The system converts the report into the chosen format: a. PDF: Lays out the report as a fixed-layout, print-ready document with charts as images. b. **Excel:** Exports the raw data and summary tables into spreadsheet sheets for further analysis. 7. The system sends the file as a download to the organizer's browser. 8. The organizer saves the file to their local machine.
Exception	1. Report not found: The system displays an error: "The requested report is not available. Please try generating it again.
Alternative Flows:	1. Share Report via Link: a. From the main reports dashboard or while viewing a report, the organizer clicks a "Share" button. b. The system generates a secure, unique, and time-limited link that provides view-only access to the report. c. The organizer can copy the link to share via email or other external communication channels. d. When a recipient opens the link, the system displays the report in a read-only web view.
Includes:	[6.1] [6.2]

Table 15 *SCEGA Authentication and Compliance use-case specification*

Use Case ID:	13		
Use Case Name:	SCEGA Authentication and Compliance		
Created By:	Abdullah Alshammari	**Last Updated By:**	
Date Created:	15/10/2025	**Date Last Updated:**	
Actors:	• Organizer		
Description:	Ensure event compliance with SCEGA regulations, otherwise reject it and notify the organizer.		
Trigger:	The organizer clicks create event button.		
Preconditions:	• The organizer has an account in Eventia system. • The organizer possesses SCEGA license/authorization credentials. • SCEGA gateway is reachable.		

Postconditions:	• **On success:** Event published and linked to the verified organizer. • **On failure:** Operation is blocked; organizer is notified.
Normal Flow:	1. Organizer submits the event request for approval. 2. System prepares event data and sends it to the SCEGA gateway. 3. SCEGA gateway validates all event details (license, venue, category, compliance documents). 4. SCEGA gateway returns an approval response to the system. 5. System publishes the event and updates its status to Approved.
Exception	**SCEGA Gateway Not Reachable** 1. System attempts to send request but fails to connect. 2. System displays an error message to the Organizer. 3. Event submission remains Pending until connection is restored. **Invalid SCEGA Credentials** 1. SCEGA gateway rejects the authorization token. 2. System blocks submission and notifies Organizer to update credentials.
Alternative Flows:	**Event Rejected by SCEGA:** 1. SCEGA gateway returns a Reject response. 2. System sets event status to Rejected. 3. System notifies the Organizer with the rejection reason. **Missing or Invalid Details:** 1. SCEGA gateway identifies missing information. System returns event to Organizer with a request for revisions.
Includes:	[7.0]

5.3 Activity Diagrams

5.3.1 Attendee Contact Organizer:

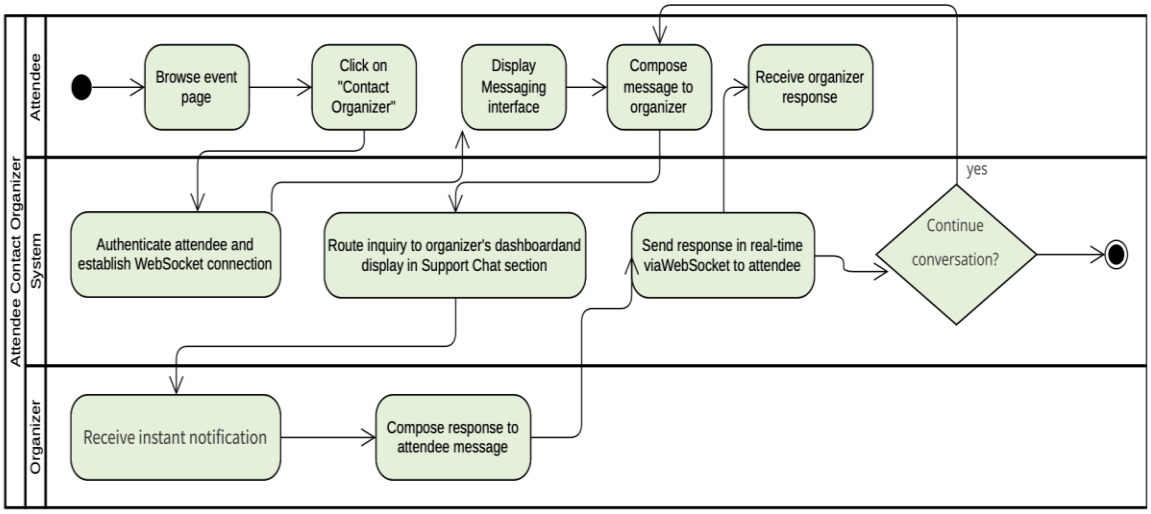

Figure 53 *Attendee Contact Organizer activity diagram*

5.3.1.1 Procedure Description:

An attendee browses an event page and initiates contact with the organizer through the messaging interface. The system authenticates the attendee, establishes a WebSocket connection, and routes the inquiry to the organizer's dashboard. The organizer receives instant notifications and can respond in real-time, enabling a continuous conversation between the attendee and organizer.

5.3.1.2 Details of Procedure Activities:

Table 16 *Details of procedure Activities for attendee contact organizer*

Activity Code	Activity	Role	Input/Trigger	Output	Description
01	Browse event page	Attendee	Navigate to event listing or event details	Event page displayed to attendee	The attendee navigates through the system and views the event page to learn more about the event details, schedule, and other information.

02	Click on "Contact Organizer"	Attendee	Attendee action on event page	Request sent to system to initiate contact	The attendee clicks on the "Contact Organizer" button or link available on the event page to initiate communication with the event organizer.
03	Authenticate attendee and establish WebSocket connection	System	Authenticate attendee and establish WebSocket connection	WebSocket connection established	The system verifies the attendee's identity and establishes a secure WebSocket connection to enable real-time bidirectional communication between the attendee and organizer.
04	Display Messaging interface	System	Successful authentication and connection	Messaging interface displayed to attendee	The system displays an interactive messaging interface where the attendee can compose and send messages to the organizer.
05	Compose message to organizer	Attendee	Text message content	Message ready to be sent	The attendee types their inquiry, question, or message in the messaging interface to communicate with the event organizer.
06	Route inquiry to organizer's dashboard and display in Support	System	Attendee's composed message	Message delivered to organizer's dashboard in Support	The system routes the attendee's message through the WebSocket connection to the organizer's dashboard and

	Chat section			Chat section	displays it in the designated Support Chat section for organizer review
07	Receive instant notification	Organizer	Incoming message from attendee	Notification displayed to organizer	The organizer receives an instant real-time notification alerting them that an attendee has sent a message, prompting them to review and respond.
08	Compose response to attendee message	Organizer	Attendee's message content	Response message composed	The organizer reviews the attendee's inquiry and composes an appropriate response message in the Support Chat interface.
09	Send response in real-time via WebSocket to attendee	System	Organizer's response message	Response delivered to attendee in real-time	The system transmits the organizer's response through the WebSocket connection, ensuring instant delivery to the attendee's messaging interface.
010	Receive organizer response	Attendee	Organizer's response message	Response displayed in messaging interface	The attendee receives the organizer's response in real-time within the messaging interface, allowing them to read the reply immediately
011	Continue conversation?	System	Current conversation state	- If yes: Return to compose	The system checks whether the attendee or organizer wishes to

				message step - If no: End conversation	continue the conversation. If yes, the messaging interface remains active for additional messages. If no, the conversation ends and the WebSocket connection may be closed

5.3.2 Creating a new Event

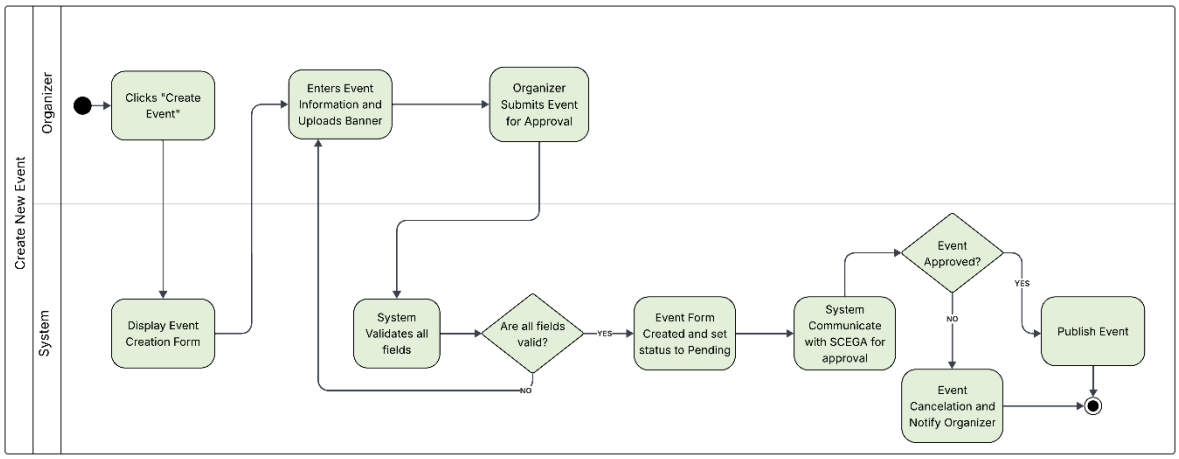

Figure 54 *Create New Event activity diagram*

5.3.2.1 Procedure Description

This procedure explains how an organizer creates a new event using the system. The organizer selects the “Create Event” option, enters essential details such as title, date, time, and location, and uploads an event banner if needed. The system validates all mandatory fields, Communicate through SCEGA Gateway for approval, then displays a confirmation message once the event is successfully created otherwise notify the Organizer.

5.3.2.2 Details of Procedure Activities

Table 17 *Details of procedure Activities for Create New Event*

Activity Code	Activity	Role	Input	Output	Description
001	Clicks "Create Event"	Organizer	User selection	System command to open event form	The organizer selects the "Create Event" option from the dashboard.
002	Displays Event Creation Form	System	User command	Event creation form displayed	The system displays a form for entering event details.
003	Enter Event Information and Uploads Banned.	Organizer	Event title, description and other details with an optional banner.	Event details submitted	The organizer enters all required event information in the form.
004	System Validates Event Fields	System	Submitted event data	Validation Status	The system checks that all mandatory fields are correctly filled.
005	Are All Fields Valid?	System	Validation result	Decision to continue or ask to re-enter valid details	The system determines whether all entered event details are valid or invalid.
006	Creates Event and set status to "Pending"	System	Validate event data	Event record created and its status becomes "Pending"	The system saves the event details into the database and Set its status to "Pending" waiting confirmation from SCEGA.
007	System Communicate with SCEGA for approval	System	Organizer Credential and Event details	Event approved or denied	The system sends Event Creation request to SCEGA Gateway

008	Event Approved?	System	SCEGA output	Decision to continue or terminate	The system determines whether event is approved or denied
009	Event Cancelation and Notify Organizer	System	Decision of Event Rejection	Organizer Notification	In case of SCEGA rejected the event, the event form is canceled and organizer notified
010	Publish Event	System	Decision of Event Approval	Publishing the Event	In case of SCEGA Approved the event, Event is published

5.3.3 Update Existing Events

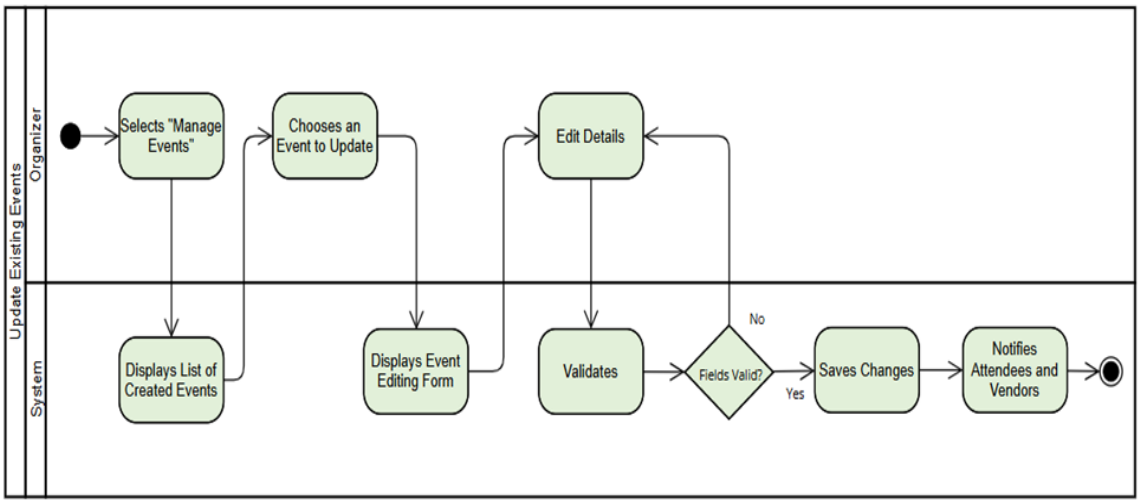

Figure 55 *Update Existing Events activity diagram*

5.3.3.1 Procedure Description

This procedure explains how an organizer updates the details of an existing event through the system. The organizer selects an event from the list, edits its information such as title, date, time, or location, and submits the changes. The system validates the updated data, saves the modifications if valid, and automatically notifies all registered attendees and vendors of the updates.

5.3.3.2 Details of Procedure Activities

Table 18 *Details of procedure Activities for Update Existing Events*

Activity Code	Activity	Role	Input	Output	Description
001	Selects "Manage Events"	Organizer	User selection	List of existing events displayed	The organizer selects the option to manage previously created events.
002	Display List of Events	System	User command	List of created events	The system displays all events created by the organizer.
003	Chooses an event to update	Organizer	User selection	System command to open event editing form	The Organizer Chooses an event to update it's details.
004	Displays Event Editing Form	System	User Command	Event Editing form displayed	The system displays a form for editing event details.
005	Edit Details	Organizer	Update event title, description, date, time, ticket price, or location	Event details submitted	The organizer edit event information in the form.
006	Validates	System	Submitted event data	Validation Status	The system determines whether all entered event details are valid or invalid.
006	Fields Valid?	System	Validation result	Decision (valid or invalid)	The system determines whether all updated event details are valid or invalid.
007	Saves Changes	System	Validate event data	Event record updated and confirmation	The system saves the event details into the database and confirms

				message displayed	successful event update to the organizer
008	Notifies Attendees and Vendors	System	Updated event details	Notification sent	The system automatically notifies all registered attendees and vendors of the update.

5.3.4 Delete Existing Events

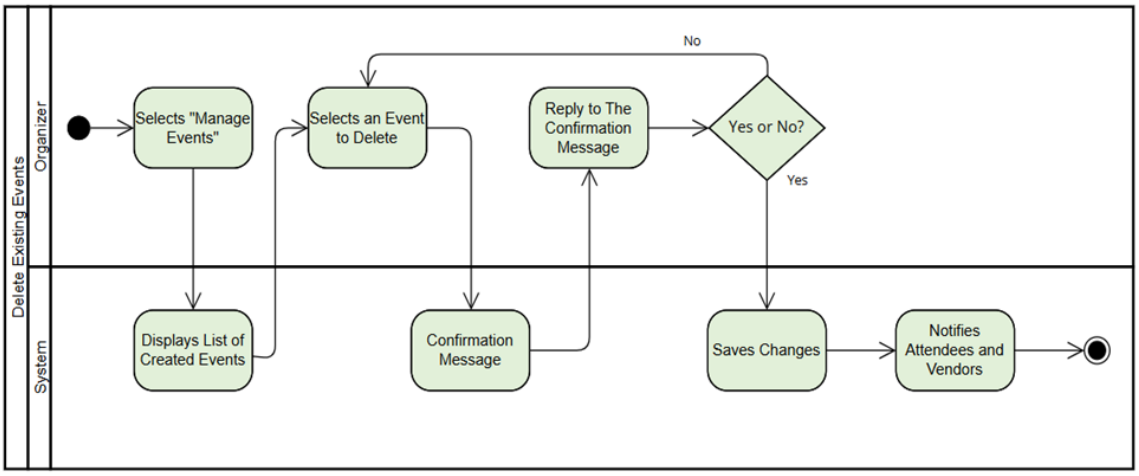

Figure 56 *Delete Existing Events activity diagram*

5.3.4.1 Procedure Description

This procedure describes how an organizer deletes an existing event from the system. The organizer selects an event from the list and chooses the delete option. The system requests confirmation before permanently removing the event from the database. Once deleted, the system automatically notifies all registered attendees and vendors about the event cancellation.

5.3.4.2 Details of Procedure Activities

Table 19 *Details of procedure Activities for Delete Existing Events*

Activity Code	Activity	Role	Input	Output	Description

001	Selects “Manage Events”	Organizer	User selection	List of existing events displayed	The organizer selects the option to manage previously created events.
002	Displays List of Events	System	User command	List of created events	The system displays all events created by the organizer.
003	Selects an Event to Delete	Organizer	User selection	System command to open event editing form	The Organizer Chooses an event to update it’s details.
004	Confirmation Message	System	User Command	Event Editing form displayed	The system displays a form for editing event details.
005	Reply to The Confirmation Message	Organizer	Update event information	Event details submitted	The organizer edits event information in the form.
006	Yes or No?	Organizer	Reply result	Decision (yes or no)	The system determines whether all entered event details are valid or invalid.
007	Saves Changes	System	Validation result	Event record deleted and confirmation message displayed	The system removes the event details from the database and confirms successful event deletion to the organizer
008	Notifies Attendees and Vendors	System	Deleted event details	Notification sent	The system automatically notifies all registered attendees and vendors of the cancellation.

5.3.5 Invite Vendors

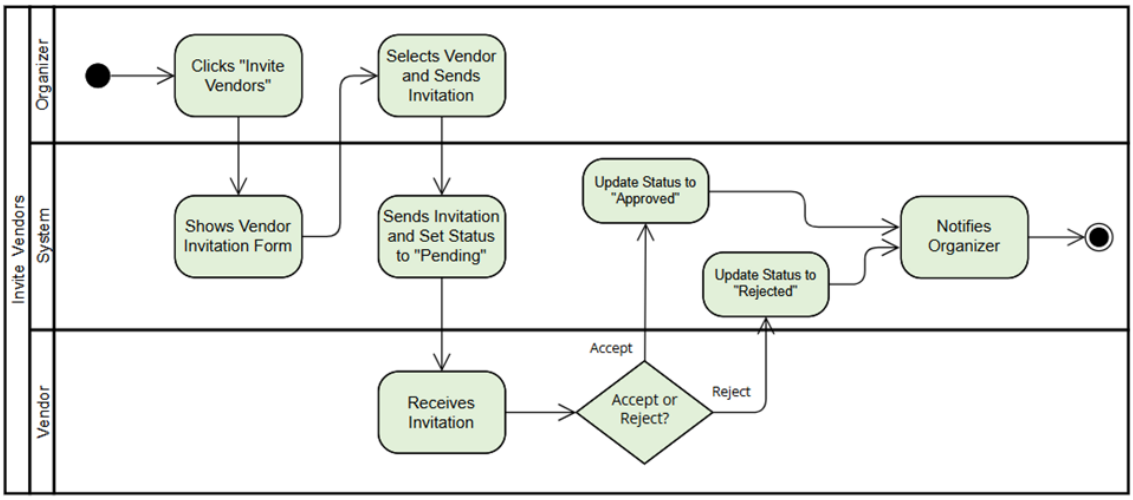

Figure 57 *Invite Vendors activity diagram*

5.3.5.1 Procedure Description

This procedure explains how an organizer invites vendors to participate in an event. The organizer selects an event and chooses the "Invite Vendors" option. The system displays a list of available vendors, allowing the organizer to send digital invitations. Once invitations are sent, vendors can respond by accepting or rejecting them, and the system records their responses for the organizer to review.

5.3.5.2 Details of Procedure Activities

Table 20 *Details of procedure Activities for Invite Vendors*

Activity Code	Activity	Role	Input	Output	Description
001	Click "Invite Vendors"	Organizer	User selection	Vendor invitation interface command	The organizer selects an event and clicks the "Invite Vendors" option.
002	Display Vendor Invitation Form	System	User command	Vendor invitation form displayed	The system displays the invitation form with available vendors and event details.

003	Select Vendors and Send Invitations	Organizer	Vendor list and event details	Vendor invitations submitted	The organizer selects one or more vendors and sends digital invitations through the system.
004	Set Invitation Status to “Pending”	System	Invitation submission	Vendor invitation status: Pending	The system records each sent invitation and sets its status to “Pending.”
005	Receives Invitation	Vendor	System notification	Invitation received	The vendor receives the digital event invitation through their dashboard or email.
006	Accept or Reject?	Vendor	Invitation details	Accept or reject decision	The vendor reviews the event information and chooses to either accept or reject the invitation.
007	Update Status to “Approved” (Decision: Yes)	System	Vendor acceptance	Invitation status updated to “Approved”	If the vendor accepts, the system updates the invitation status to “Approved.”
008	Update Status to “Rejected” (Decision: No)	System	Vendor rejection	Invitation status updated to “Rejected”	If the vendor rejects, the system updates the invitation status to “Rejected.”
009	Notifies Organizer	System	Vendor response	Notification sent	The system notifies the organizer of the vendor’s response and

					updates the event's vendor list accordingly.

5.3.6 Reports and Feedback

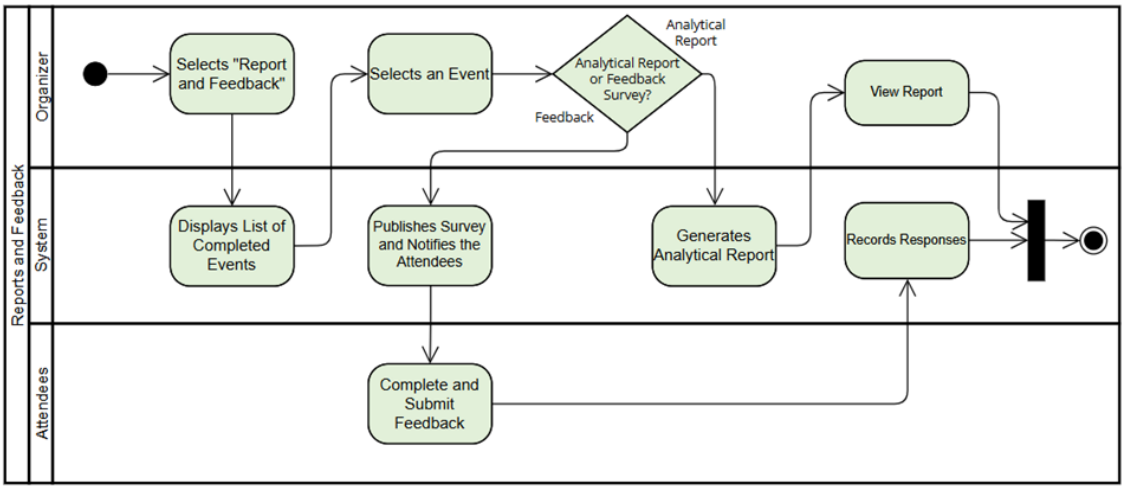

Figure 58 *Reports and Feedback activity diagram*

5.3.6.1 Procedure Description

This procedure explains how an organizer manages post-event reports and feedback. The organizer selects a completed event to either view its analytical report or create a feedback survey. If a survey is published, attendees are notified and can submit their feedback. The system records all responses and combines them with the analytical report for the organizer to review.

5.3.6.2 Details of Procedure Activities

Table 21 *Details of procedure Activities for Reports and Feedback*

Activity Code	Activity	Role	Input	Output	Description
001	Select "Reports and Feedback"	Organizer	User Selection	Reports and feedback module accessed	The organizer selects the "Reports and Feedback" option from the dashboard.
002	Display List of Completed Events	System	User command	List of completed events displayed	The system displays all events that have

					been marked as completed.
003	Select Event	Organizer	Completed event list	Event selected	The organizer selects a completed event to view its report or manage feedback.
004	Analytical Report or Feedback Survey?	Organizer	Selected event	Selected action (Report or Feedback)	The organizer chooses whether to view the analytical report or manage post-event feedback.
005	Publishes Survey and Notifies the Attendees (Decision: Feedback Survey)	System	Survey questions and event selection	Feedback survey published and attendees notified	The System publishes a post-event feedback or complaint survey, and notifies attendees that the survey is available for completion.
006	Complete and Submit Feedback	Attendee	Survey form	Feedback responses submitted	Attendees complete and submit the post-event feedback survey.
007	Record Responses	System	Submitted feedback	Responses stored and linked to event	The system records all submitted feedback and links it to the corresponding event record.
008	Generates Analytical Report (Decision: Analytical Report)	System	Completed event data	Analytical report generated	If “Analytical Report” is chosen, the system compiles event data such as attendance, vendor participation, and feedback

					results into a report.
009	View Report	Organizer	Generated report	Report displayed	The organizer views the generated analytical report through the reports dashboard.

5.3.7 SCEGA Authentication and Compliance Specification

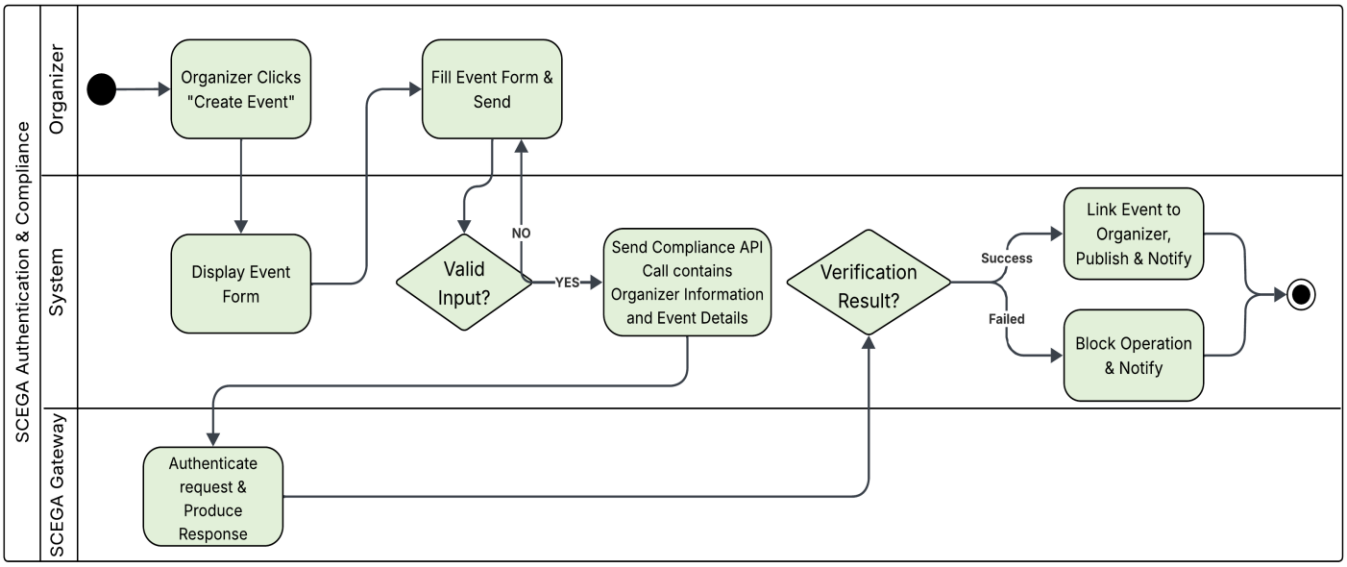

Figure 59 *SCEGA Authentication and Compliance Specification activity diagram*

5.3.7.1 Procedure Description

This procedure describes the process of verifying an event organizer through the SCEGA gateway before event creation and publication. The system ensures that only verified organizers with valid event details can create and publish events.

5.3.7.2 Procedure Activities Details

Table 22 *Details of Procedure Activities for SCEGA Authentication and Compliance Specification*

Activity Code	Activity	Role	Input/Trigger	Output	Description
01	Click Create Event	Organizer	Create Event Action	Event form displayed	The system presents the event creation form.
02	Display Form	System	Request Received	Event form ready	The system presents the

					event creation form.
03	Validate input	System	Submitted Form	Valid/invalid flag	The system checks for completeness and validity.
04	Send Authentication Request	System / SCEGA Gateway	Organizer and Event data	Verification response	The system sends data to SCEGA for compliance check.
05	Process Verification	SCEGA Gateway	API request	Response sent	SCEGA verifies credentials and returns result.
06	Handle Response	System	Verification result	Success or Failure	If successful, event is published; if failed, organizer is notified.

5.4 System Architecture

The system architecture is organized into multiple layers to adapt a modern design where the team can develop independent components simultaneously, easier debugging and maintenance. The Front-End layer is the presentation layer where the users interact with, it provides graphical interfaces, navigation flows, and input handling components. Underneath it, the Back-End layer which encapsulates the application's core logic and users APIs. The Database Connection layer acts as intermediary that connects the Back-End and the system Database which stores event records, users, vendor information, feedback logs, and related system data.

5.4.1 System Architecture Description

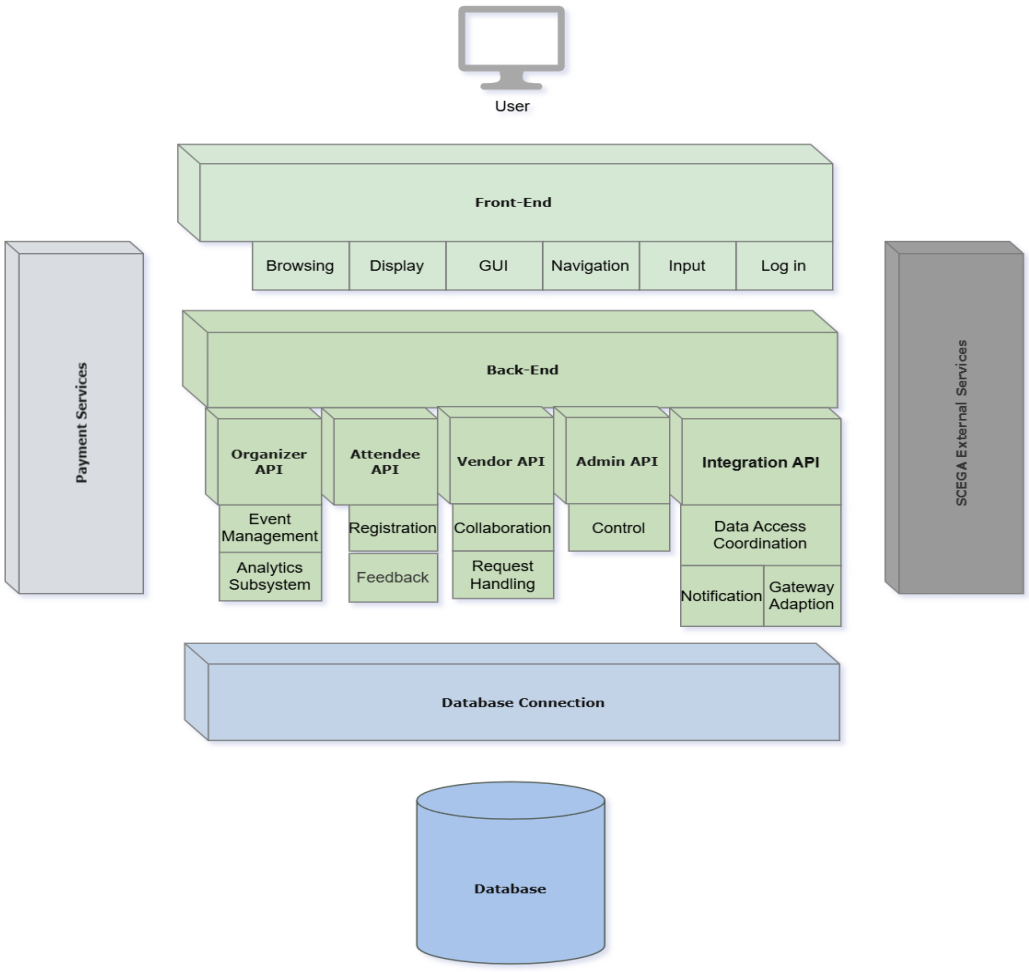

Figure 60 *Eventia System Architecture*

User: represents anyone who uses the system such as organizers, vendors, attendees, and administrators. Each user type has its own role and set of functionalities that satisfies the user's needs.

Front-End: is the part of the system that users see and interact with directly. It is responsible for how the system looks and how users move through it. It handles all visual design, navigation, and input. The front-end has several components: browsing, display, GUI, navigation, input, login and other minor services.

Browsing: allows users to explore public events and view events' details. This feature allows simplicity and ease of use for the users where they can check events before logging in or even signing up to the system.

Display: it's the concept where different information and notifications take a place in the Front-End layer, such as Error messages, Confirmations and other types.

Graphical User Interface (GUI): provides a clean and easy layout that users can understand quickly. It includes buttons, forms, and icons that help users perform actions such as creating events, booking tickets, or updating profiles.

Navigation: helps users move smoothly between pages and sections. It allows users to go from the home page to dashboards, event lists, feedback page, and reports without confusion.

Input: is where users provide data to the system, such as filling out forms, checking for log in data against the log in policy or submitting feedback. The system checks each input to make sure it's valid before sending it to the back-end.

Log In: the section that lets users access their accounts based on their role. It checks their credentials and loads their specific needs in the homepage, such as organizer, vendor, or attendee. Guests skip this step and can still browse public events and even get their ticket.

Back-End: is the main engine of the system. It handles logic, processes user requests, and communicates with the database. It includes several APIs that manage the actions of different user types. Each API focuses on the needs of a particular user group or system task.

Organizer API: lets organizers create and manage their events, they also have the ability to edit, and delete events. It also helps them invite vendors, collect feedback, and view analytics about event success. This API gives organizers the full control over managing events and tracking their performance.

Attendee API: handles registration and feedback for people who attend events. It allows attendees to register for events, view their booked ones, and share their opinions afterward. Attendees can also edit their profile information, such as updating their picture, username, or other personal details.

Vendor API: manages vendor participation and communication with organizers. It lets vendors view and respond to invitations that they get from the organizers, apply to participate in a specific event, manage their profiles, and track events they are part of. This API helps vendors collaborate efficiently and stay updated.

Admin API: allows administrators to control and monitor the whole platform. They can manage users, check activities, and keep the system secure. It makes sure that everything runs properly without errors or misuse.

Integration API: connects Eventia to outside systems like payment gateways and the SCEGA verification system. It helps with sending notifications, verifying organizers before events go public, and managing external data exchanges. This keeps the platform connected and reliable.

Database Connection: acts as the bridge between the back-end and the main database. It make sure that all information, such as user details, event data, and feedback, moves correctly and securely between the two layers.

Database: is where all system information is stored. It keeps records of users, events, vendors, attendees, and feedback. Whenever someone registers, submits data, or updates information, it's saved here for future use. The database ensures that everything remains organized and accessible.

Payment Services: handles all the system's financial actions. They make sure that event payments, vendor fees, and attendee transactions are processed safely and recorded correctly. This allows for smooth and secure payment operations between all users.

SCEGA External Services: is the component that connects Eventia to SCEGA for gaining the official permission after authentication and compliance phases passing. SCEGA verifies the identity of organizers before they can publish events, also compares the event details against The Kingdom regulations. This helps the events in The Kingdom stay consistent and trustworthy.

5.5 Database Design

This section presents the core entities that form the foundation of the system's database structure. Each entity represents a key user or component within the platform, along with the attributes necessary for identification, authentication, communication, and interaction. The defined entities Organizer, Vendor, Attendee, Guest, Event, Report, and Feedback capture essential information that supports the system's main functions, such as event creation, service management, user participation, ticket handling, and performance tracking. By outlining the purpose and role of each attribute, this section establishes a clear understanding of how data is organized and how different users and processes are connected throughout the system.

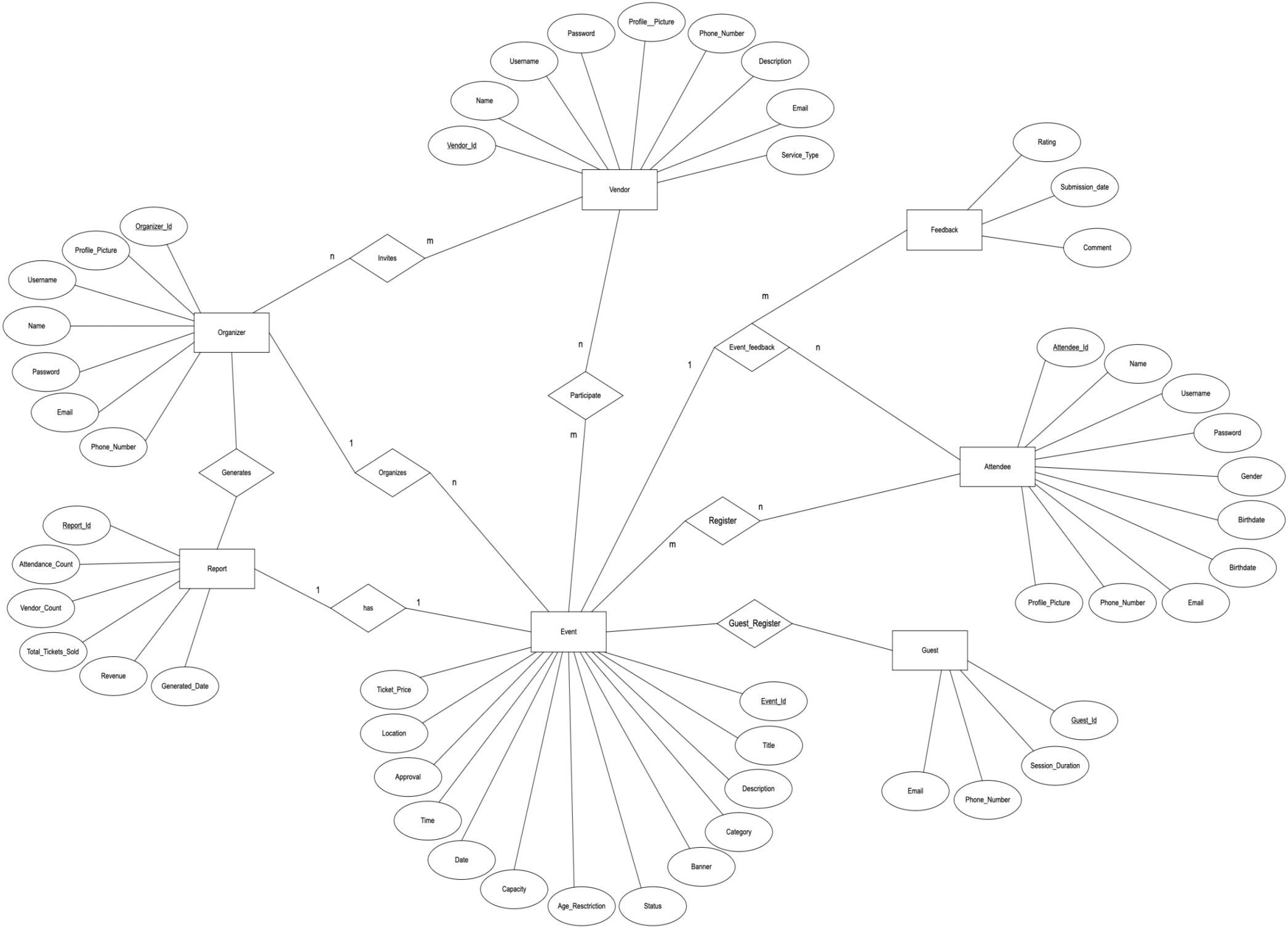

Figure 61 *Eventia Database design*

5.5.1 Entity Descriptions

Organizer(Organizer_ID, Username, Name, Password, Email, Phone_Number, Profile_Picture)
Organizer_ID: a unique identifier used to distinguish each organizer in the system.
Username: the name the organizer uses to log in and be recognized in the platform.
Name: the organizer's name as displayed in their profile and on the events they manage.
Password: the organizer's secure login password for accessing their account.
Email: the organizer's email address used for login and communication.
Phone_Number: the organizer's contact number for verification or event-related communication.
Profile_Picture: the organizer's displayed profile image.

Vendor(Vendor_ID, Username, Name, Password, Email, Phone_Number, Service_Type, Profile_Picture, Description)
Vendor_ID: a unique identifier assigned to each vendor.
Username: the name the vendor uses to log in and be recognized in the system.
Name: the vendor's business name displayed in their profile.
Password: the secure password the vendor uses to log into the system.
Email: the vendor's email address used for login and communication.
Phone_Number: the vendor's contact number for event coordination.
Service_Type: the type of service the vendor provides.
Profile_Picture: the vendor's displayed profile image.
Description: a short introduction or bio where the vendor explains their service or background.

Attendee(Attendee_ID, Name, Username, Password, Email, Phone_Number, Gender, Birthdate, Preferences, Profile_Picture)
Attendee_ID: a unique identifier assigned to each attendee.
Name: the attendee's full name as shown in their profile.
Username: a custom name the attendee uses within the platform.
Password: the attendee's secure login password.
Email: used for login, communication, and event notifications.
Phone_Number: attendee's contact number for updates or reminders.
Gender: the attendee's gender (optional and used for personalization or analytics).
Birthdate: the attendee's date of birth, used for age verification or tailored suggestions.
Preferences: categories or event types the attendee is interested in.
Profile_Picture: the attendee's profile image displayed on their account.

Guest(Guest_ID, Phone_Number, Email, Session_Duration)
Guest_ID: a unique identifier for each guest user, used to track their actions even though they do not log in or register.
Phone_Number: an optional contact number the guest can provide to receive event tickets or updates without creating an account.
Email: an optional email address guests can use instead of a phone number to get their ticket and notifications.

Session_Duration: the amount of time the guest spends browsing or interacting with the platform during their visit.

Event(Event_ID, Title, Description, Status, Category, Capacity, Age_Restriction, Date, Time, Approval, Location, Ticket_Price, Banner)
Event_ID: a unique identifier used to distinguish each event in the system.
Title: the event's name as displayed to users.
Description: a detailed explanation of what the event is about.
Status: indicates whether the event is upcoming, completed, or cancelled.
Category: the type of event.
Capacity: the maximum number of people who can attend the event.
Age_Restriction: the minimum age required to attend the event, if any.
Date: the calendar date when the event will take place.
Time: the time the event is scheduled to start.
Approval: shows whether the event is approved, pending, or rejected by SCEGA
Location: the physical or online place where the event will be held.
Ticket_Price: the cost of attending the event (or zero for free events).
Banner: an image used to visually represent and promote the event.

Report(Report_ID, Attendee_Count, Generated_Date, Vendor_Count, Total_Tickets_Sold, Revenue)
Report_ID: a unique identifier assigned to each generated report.
Attendee_Count: the total number of attendees who participated in the event.
Generated_Date: the date when the report was created.
Vendor_Count: the number of vendors involved in or assigned to the event.
Total_Tickets_Sold: the number of tickets purchased or claimed by users.
Revenue: the total income generated from ticket sales.

Feedback(Submission_Date, Comment, Rating)
Submission_Date: the date when the attendee submitted their feedback for the event.
Comment: the attendee's written opinion or message about their experience.
Rating: a score given by the attendee (such as 1–5 stars).

5.6 User Interface design

This chapter shows selected user interface screens of the Eventia system. These screens explain how users can interact with the system and access some of its main features in a simple and organized way.

5.6.1 Sign Up Page

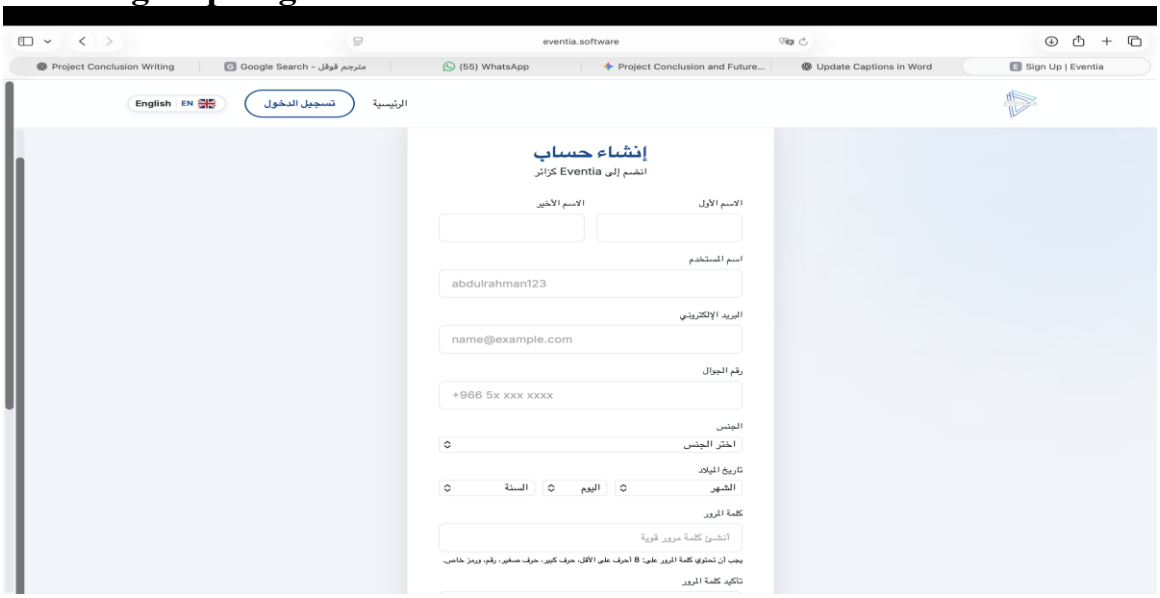

Figure 62: *Attendee sign up page*

5.6.2 Attendee Home Page

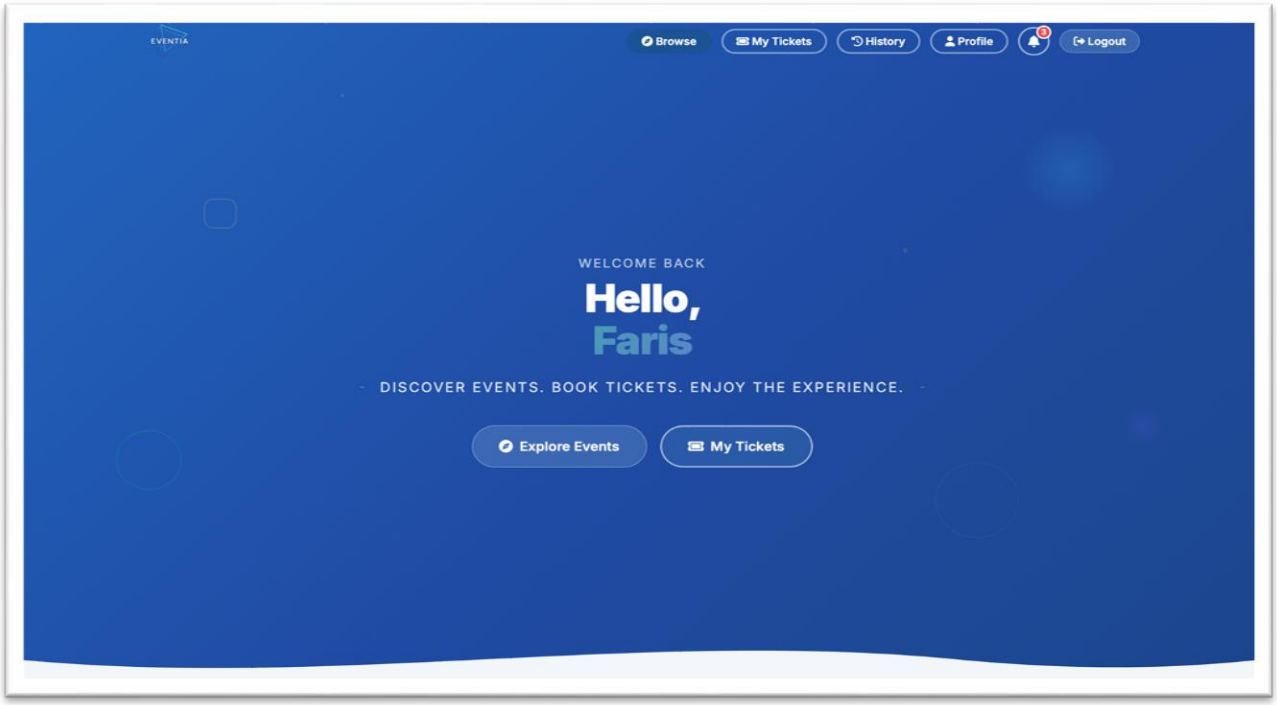

Figure 63: *Attendee Home Page*

5.6.3 Attendee Browse Events

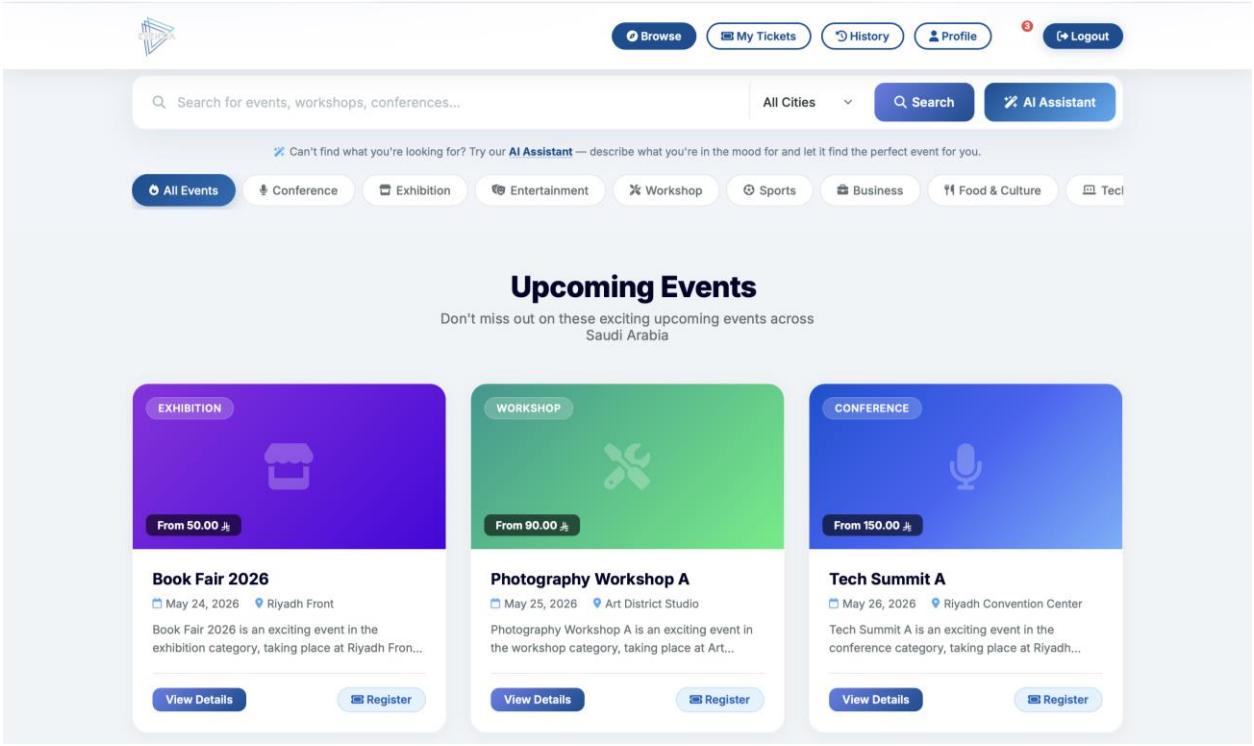

Figure 64: *Browse Events Page*

5.6.4 Organizer Dashboard

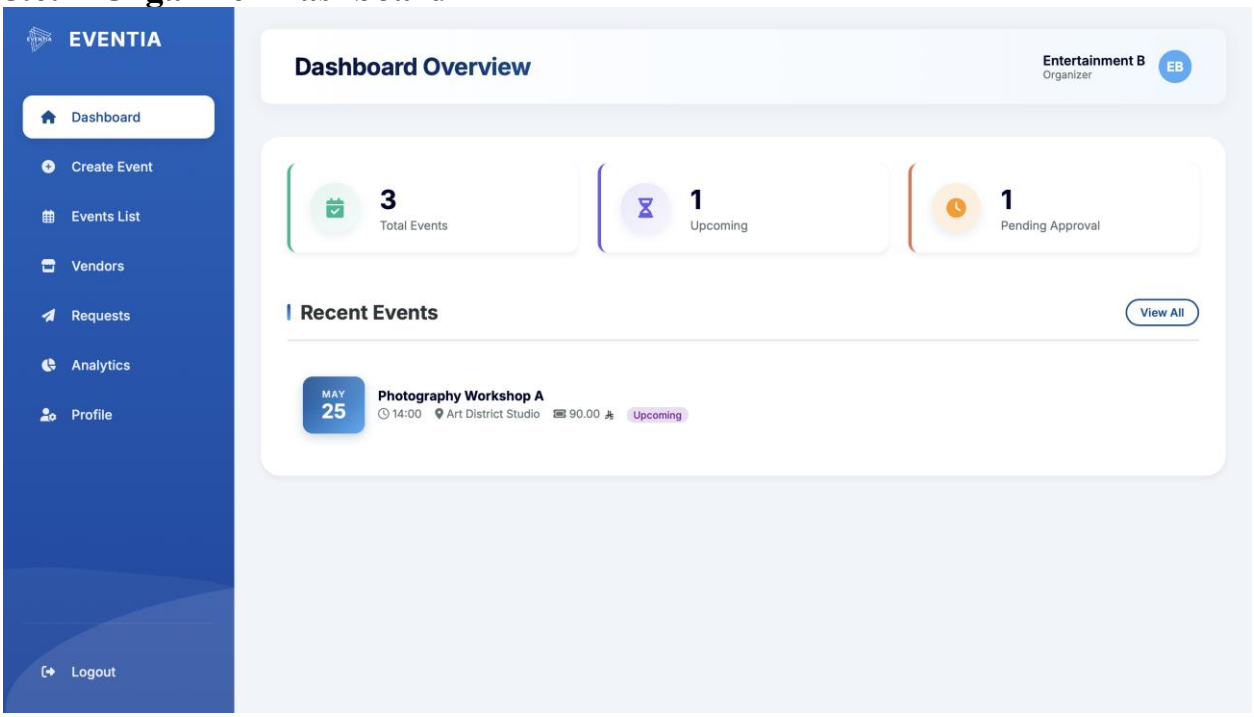

Figure 65: *Organizer Dashboard*

5.6.5 Organizer Event Analytics

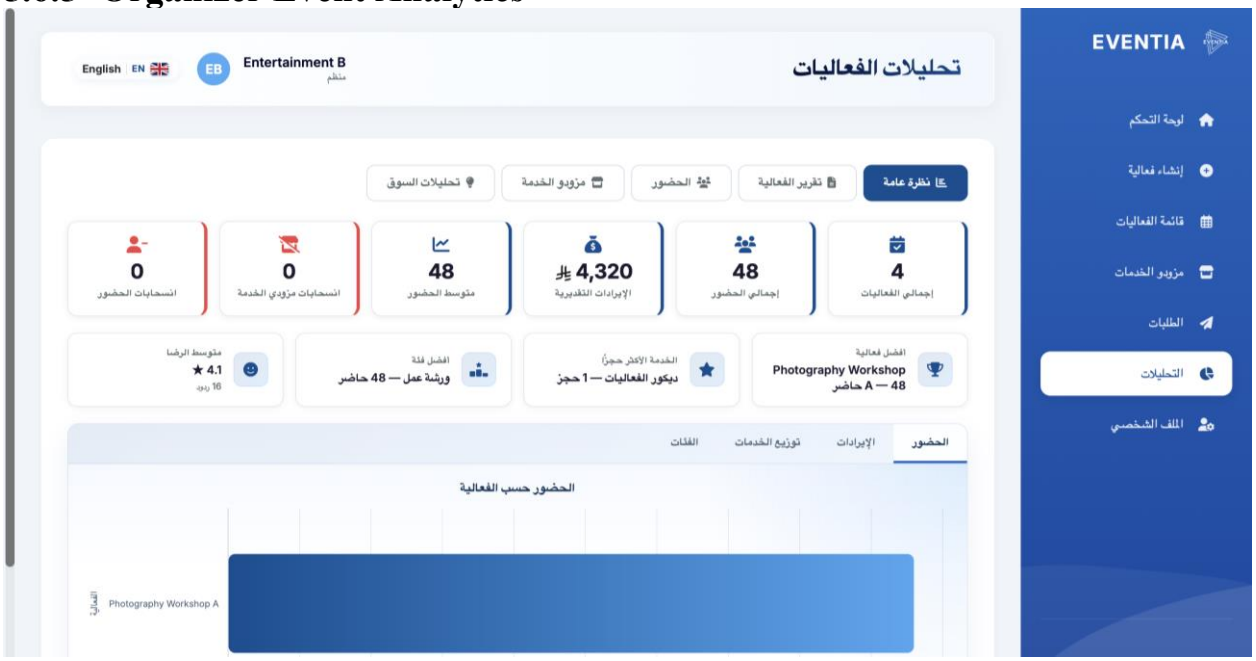

Figure 66: *Organizer Event Analytics*

5.6.6 Event Management

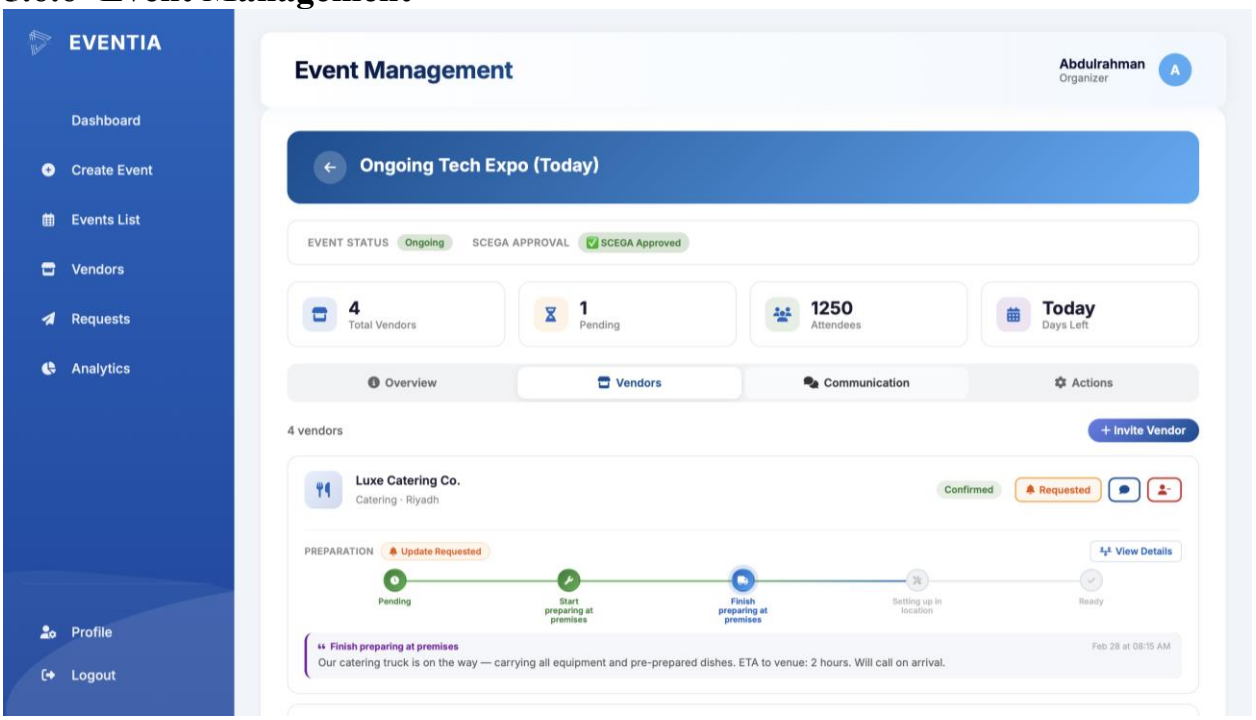

Figure 67*: Event Management for Organizer*

5.6.7 Vendor Dashboard

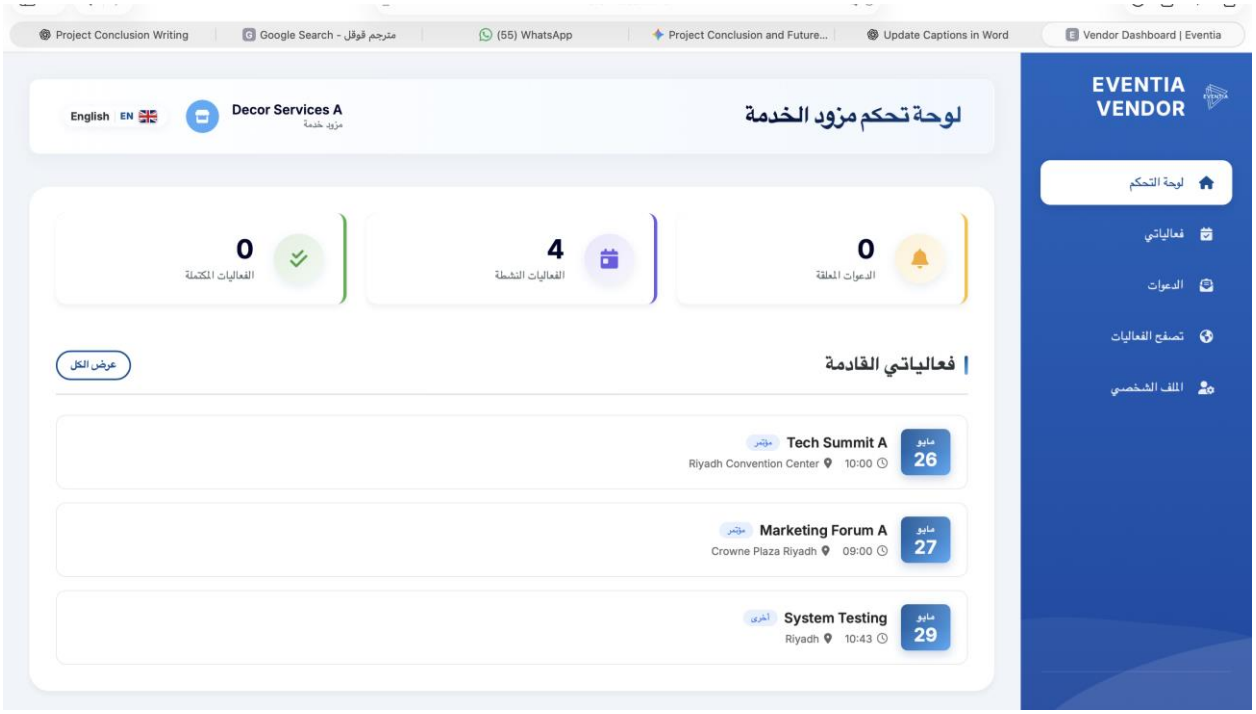

***Figure 68**: Vendor Dashboard*

5.6.8 Vendor Browse Events

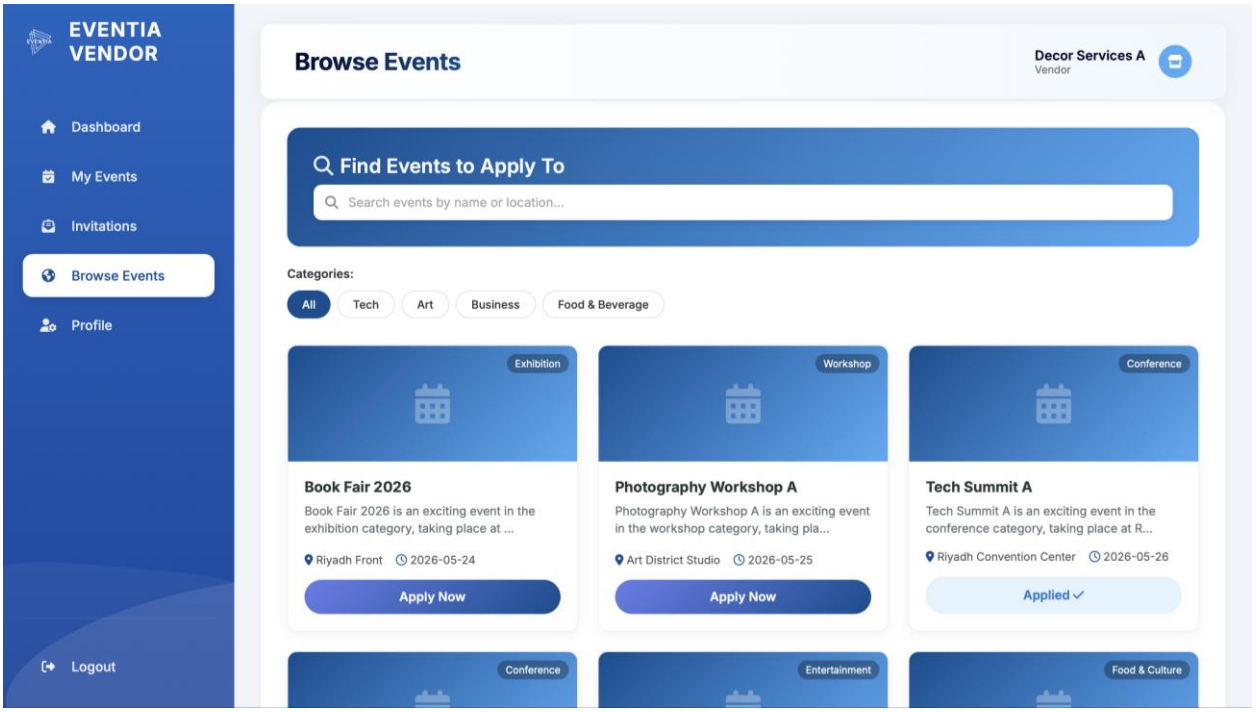

***Figure 69**: Vendor Browsr Events*

Chapter 6: Implementation

In this chapter, we will discuss the tools and technologies that we have used to develop our web platform. Building on the requirements defined in Chapter 4 and the design presented in Chapter 5, this chapter focuses on how those decisions were translated into a working system starting from development tools and frameworks to implementation details.

6.1 Tools and Frameworks

6.1.1 UI/UX Design

For designing the user interface of Eventia, we utilized Claude Opus 4.6 integrated within the Cursor and Antigravity development environment as our primary prototyping tool. Rather than using traditional design software, we leveraged prompt engineering techniques to iteratively generate and refine the front-end prototype. This approach allowed us to rapidly explore different layout options, color schemes, and component structures through natural language descriptions. By progressively enhancing the prompts with more specific requirements—such as role-based dashboard layouts, responsive navigation patterns, and consistent design tokens—we were able to produce a cohesive and polished user interface that met our design goals. The iterative prompt engineering process enabled us to move quickly from concept to a functional prototype without the overhead of traditional design-to-code handoff workflows.

6.1.2 Front-End

The front-end of Eventia was built using HTML5, CSS3, and JavaScript in the Cursor and Antigravity development environments, where Claude Opus 4.6 was used to assist in generating and refining the front-end code. The implementation was completed without relying on front-end frameworks such as React or Vue, which provided complete control over performance, styling, and the overall user experience. The application follows a server-rendered architecture using Django templates, with client-side JavaScript handling dynamic interactions within each dashboard.Google

Fonts:

Google Fonts is a free, open-source font library that provides a wide selection of web-optimized fonts. For Eventia, we selected Inter as the primary typeface across all pages. Inter is a modern, highly legible sans-serif font designed specifically for computer screens, loaded with weights ranging from 300 (Light) to 900 (Black) to support diverse typographic needs—from subtle labels to bold dashboard headings.

Font Awesome:

Font Awesome 6.4.0 was used throughout the platform to enhance the interface with clear, universally recognizable icons. Icons appear in sidebar navigation, action buttons, stat cards, notification badges, and category labels. Using a consistent icon library

improves the user experience by providing immediate visual cues and reducing cognitive load when navigating between different sections of the platform.

Chart.js:

For data visualization on the organizer dashboard, we used Chart.js loaded via the jsDelivr CDN, along with the chartjs-plugin-datalabels extension for inline data labels. The analytics page renders interactive charts covering attendance by event, revenue breakdowns, service distribution, category performance, and age demographics. All chart data is injected from the Django back-end through template variables, minimizing additional HTTP requests. Chart interactivity—such as switching between tabs for Attendance, Revenue, Services, and Categories—is implemented through lightweight vanilla JavaScript functions. Additionally, html2canvas and jsPDF libraries are integrated to allow organizers to export their analytics dashboards as PDF reports directly from the browser.

JsBarcode:

On the attendee dashboard, JsBarcode (version 3.11.5) is used to generate ticket barcodes dynamically on the client side. When attendees view their registered tickets, a barcode is rendered for each ticket, enabling easy verification at event check-in.

CSS Architecture:

Our CSS architecture is organized into nine dedicated stylesheets. Each dashboard has its own CSS files—the organizer dashboard, vendor dashboard, and attendee dashboard each have a dedicated stylesheet that handles their unique layouts and components independently. A shared base stylesheet defines a unified design system using CSS custom properties as reusable tokens, and a common dashboard stylesheet provides the sidebar and header structure used across all roles. Separate stylesheets also serve the authentication pages, the public landing page, and the AI assistant overlay. For responsiveness, we manually applied CSS media queries at multiple breakpoints using CSS Grid and Flexbox, ensuring optimal rendering across desktops, tablets, and smartphones.

JavaScript Architecture:

All client-side logic is written in JavaScript without any framework dependencies. Each user role has a dedicated logic file—organizer-logic.js, vendor-logic.js, and attendee-logic.js—that manages dashboard navigation, data fetching via the Fetch API, form handling, and modal interactions. Dashboard sections are toggled through a view-switching pattern where sidebar navigation links use data-view attributes to control which content sections are displayed. The landing page employs an IntersectionObserver for scroll-reveal animations and implements client-side event filtering with tokenized search and category multi-select. CSRF tokens are managed through Django template injection (window.CSRF_TOKEN) and passed as headers in all Fetch API requests to maintain security.

6.1.3 Back-End

For the back-end, we used the Django Framework (version 5.2), a high-level Python web framework that ships with an ORM, an authentication system, a templating engine, an admin interface, security middleware, and a migration system out of the box. This breadth let us focus on Eventia's domain logic, events, vendors, licensing, and tickets, rather than reinventing infrastructure.

Authentication is built on Django's built-in framework, extended with a custom user model (CustomUser) that adds a role field with four choices: ORGANIZER, VENDOR, ATTENDEE, and SCEGA_ADMIN. Login is session-based, and protected views are guarded with the @login_required decorator. After authentication, the dashboard_view function inspects request.user.role and dispatches each user to their role-specific dashboard, centralizing role enforcement in a single place.

Passwords are never stored in plain text; Django hashes them automatically using PBKDF2 with SHA-256 and a unique per-user salt.

Beyond password hashing, we rely on several built-in Django security features: CSRF protection on every POST form, the @login_required decorator on every protected view, role-based authorization checks at the top of each view, and input validation through Django Forms and ModelForms. Sensitive credentials, the Django SECRET_KEY, the Resend email API key, and the Gemini API key, are loaded from environment variables in production rather than hardcoded in the source.

Rather than adopting Django REST Framework, we built JSON APIs by hand using Django's lightweight JsonResponse. Each dashboard pulls its data from a small, dedicated endpoint that serializes only the fields the relevant front-end needs. This kept the dependency surface small and the data shape tightly controlled, at the cost of writing the serialization explicitly.

6.1.4 Database Management System

We used MySQL as our primary database management system, provisioned through PythonAnywhere's managed MySQL service. MySQL is a mature, widely supported relational database with strong tooling and excellent integration with Django through the mysqlclient driver. It handles all of Eventia's persistent state, including user accounts, events, requests, tickets, messages, broadcasts, and vendor status updates, with full ACID guarantees.

Through Django's Object-Relational Mapper (ORM), we interact with the database using Python classes and queryset methods rather than raw SQL. This makes data access expressive, keeps the codebase portable should we ever switch database engines, and lets Django manage schema evolution through migrations. Each migration file records a single schema change since project inception, so the database can be rebuilt deterministically on any environment by running python manage.py migrate.

The connection is configured in Django's settings.py file through the DATABASES dictionary, where we specify the engine, database name, credentials, host, and port. In production these values are loaded from environment variables rather than committed to the source.

6.1.5 AI Assistant: Google Gemini

A distinguishing feature of Eventia is its built-in AI assistant, accessible from the public landing page. The assistant helps users discover events, answers specific questions such as "who's organizing this?" or "are there free events this weekend?", and surfaces relevant event cards directly inside the chat UI.

Under the hood we use Google's Gemini API (gemini-3.1-flash-lite) through the google-generativeai Python client. For each chat request, the back-end first validates the JSON body, then builds a compact text representation of every APPROVED event (title, category, date and time, location, price, capacity, age restriction, organizer, vendors, and a truncated description), and finally sends that representation as a system instruction together with the user's message and the last six turns of conversation history.

Two performance optimizations make this practical at scale. First, the event-context string is cached in Django's cache framework under the key ai_events_context_v1 with a 5-minute TTL, so the expensive database fan-out runs only on a cache miss. Second, we wire a post_save signal on the Event model so that whenever any event is created, edited, or has its approval status changed, the cache is invalidated immediately. The assistant never serves stale information, but it also avoids hitting the database on every keystroke.

The model is configured to return JSON directly, conforming to a strict response schema: { "reply": "...", "event_ids": [...] }. The front-end parses the reply, renders any returned event IDs as clickable event cards, and falls back gracefully if Gemini ever returns an unparseable response.

6.1.6 OTHER Tools

We used Visual Studio Code and Cursor as our primary development environments. VS Code provided a stable, lightweight editor enhanced by its Python and Django Snippets extensions, IntelliSense, integrated terminal, and Git integration, all of which significantly improved our productivity. Cursor extended this workflow with built-in AI-assisted coding capabilities, which we relied on for accelerated refactoring, inline explanations of unfamiliar code, and rapid scaffolding of repetitive Django boilerplate.

For version control we used Git and GitHub. The repository tracks every change to the codebase, and we adopted a branch and pull-request workflow for non-trivial features. GitHub also serves as our off-site backup and as the source of truth from which the production server pulls.

The platform is deployed on PythonAnywhere and reached publicly through our custom domain eventia.software via DNS CNAME records. PythonAnywhere provides the WSGI server, the MySQL instance, scheduled tasks, and static-file serving from a single managed dashboard, which kept operations simple throughout the project.

Transactional email, covering account-related notifications and password recovery, is sent through Resend over SMTP. The configuration lives in settings.py while the API key itself is loaded from the RESEND_API_KEY environment variable so it never reaches the repository.

6.2 Implementation Details

The implementation follows the Django MVT (Model, View, Template) architecture, Django's adaptation of the classical MVC pattern. The Model layer is defined in core/models.py and represents the database schema; the View layer in core/views.py contains request handlers and business logic; the Template layer in core/templates/core/ produces the rendered HTML returned to the browser. The URL router (core/urls.py) wires HTTP requests to views, and Django's built-in middleware stack handles cross-cutting concerns like authentication, sessions, and CSRF protection.

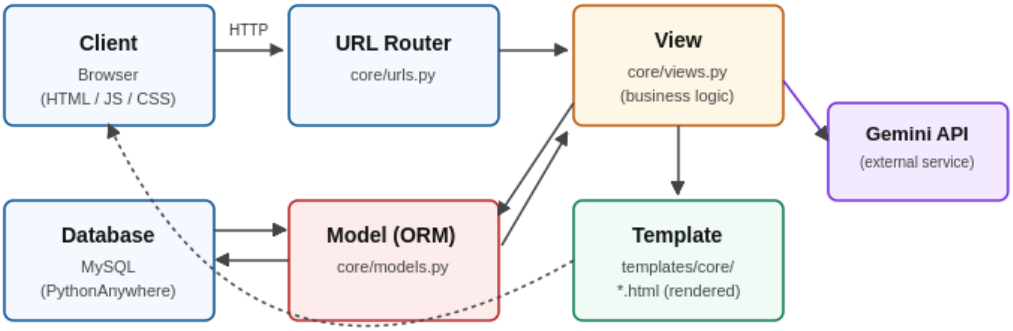

Figure 70: *Django MVT architecture as applied in Eventia*

6.2.1 Project Structure

The project is organized into two top-level Python packages: eventia_project, which contains the configuration for the Django site as a whole, and core, which is the single Django application where all of Eventia's business logic lives. Keeping all of the domain logic inside one app simplifies imports and migrations for a project of this scope, while leaving room to split the app later if the codebase grows.

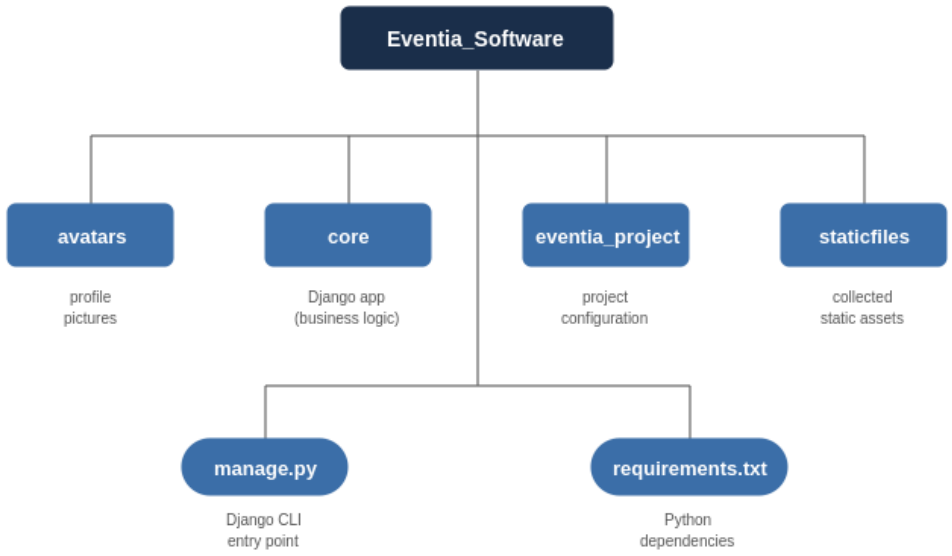

Figure 71: *Top-level project structure*

The avatars/ directory at the root stores user-uploaded profile pictures, and staticfiles/ holds the collected static assets that PythonAnywhere serves directly without going through Django.

6.2.2 Configuration: eventia_project/

The eventia_project package contains the four files Django expects for any project: settings.py (configuration), urls.py (root URL router), and wsgi.py with asgi.py (deployment entry points). Settings of note include the custom AUTH_USER_MODEL pointing at core.CustomUser, the MySQL DATABASES configuration with strict SQL mode, the Resend SMTP email backend, and environment-variable lookups for SECRET_KEY, RESEND_API_KEY, and GEMINI_API_KEY. The root URL configuration in eventia_project/urls.py is intentionally minimal: it routes /admin/ to Django's built-in admin and delegates everything else to core/urls.py, which keeps the project-level configuration generic and makes the core app self-contained.

6.2.3 The core Application

The core/ folder contains the actual application: models, views, URLs, forms, signals, the admin registration, the templates, and the static assets. Its layout follows Django conventions, so that anyone familiar with the framework can navigate the codebase without a map.

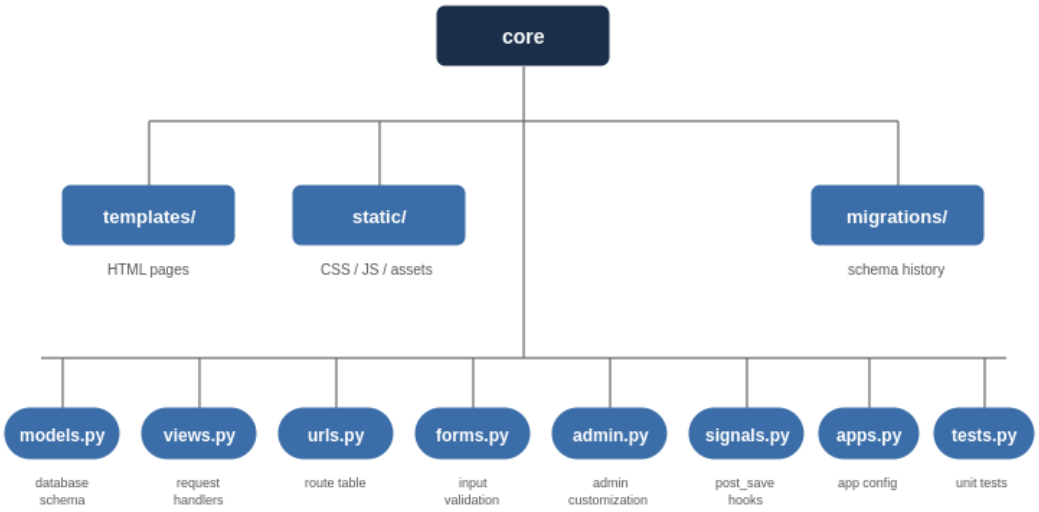

Figure 72: *core/ application structure*

6.2.4 models.py

models.py defines the structure of the database. Each class is a Django model, a Python class that Django's ORM maps to a database table. Fields map to columns, and relationships are expressed through ForeignKey and OneToOneField. The schema centres on a single CustomUser model (extending Django's AbstractUser) that adds a role field driving authorization throughout the application; each role then has a one-to-one profile model holding role-specific data.

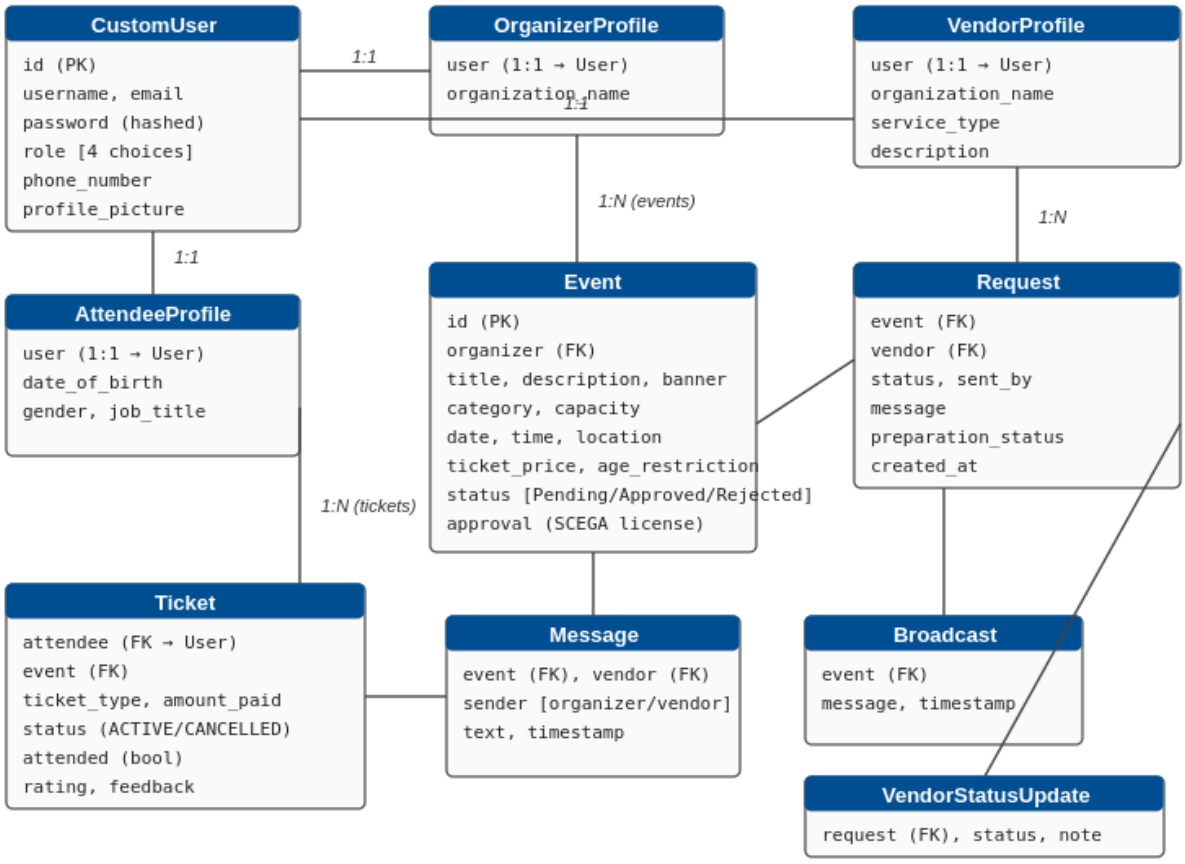

Figure 73: *Eventia data model, entities and relationships*

The custom user model is the foundation of the authentication and authorization layer. By overriding AUTH_USER_MODEL early in the project (in migration 0001), we are free to add fields like role and profile_picture without monkey-patching the default User.

```
# core/models.py

class CustomUser(AbstractUser):
    ROLE_CHOICES = (
        ('ORGANIZER',   'Organizer'),
        ('VENDOR',      'Vendor'),
        ('ATTENDEE',    'Attendee'),
        ('SCEGA_ADMIN', 'SCEGA Admin'),
    )
    role = models.CharField(max_length=20, choices=ROLE_CHOICES,
                            default='ATTENDEE')
    phone_number    = models.CharField(max_length=15, blank=True, null=True)
    profile_picture = models.ImageField(upload_to='avatars/',
                                        null=True, blank=True)
```

Figure 74: *CustomUser model, the foundation of role-based access*

The Event model captures everything about a published event: its title, description, banner image, capacity, ticket price, age restriction, and two parallel status fields. The status field tracks the publication state (PENDING, APPROVED, REJECTED), while

approval tracks the SCEGA licensing state. Splitting these makes it possible, for instance, for an event to be administratively rejected for one reason while keeping a separate audit trail of licensing decisions.

Supporting models capture the rest of the domain. The Ticket model represents an attendee's registration for an event and stores both pricing and post-event feedback fields (rating, feedback). The Request model represents collaboration requests sent between organizers and vendors, and carries a preparation_status field that vendors update as they progress from "Pending" through "Preparing", "In Transit", "Setting Up", and "Ready". Message and Broadcast support the in-app communication features, and VendorStatusUpdate gives us an audit trail of every preparation-status change.

6.2.5 forms.py

forms.py defines the input validation layer. The central piece is SignUpForm, a ModelForm that extends Django's defaults to handle a unified signup flow for all three public roles (Attendee, Organizer, Vendor). The form collects user fields and then, depending on the chosen role, also collects organisation name, service type, or date-of-birth. Validation in clean() checks that the two password fields match and that organisations supply an organisation name. In save() the password is hashed with set_password() before persisting, and the matching profile object (OrganizerProfile, VendorProfile, or AttendeeProfile) is created in the same transaction.

```
# core/forms.py  (essentials)

class SignUpForm(forms.ModelForm):
    password         = forms.CharField(widget=forms.PasswordInput)
    confirm_password = forms.CharField(widget=forms.PasswordInput)
    role             = forms.ChoiceField(choices=User.ROLE_CHOICES)

    class Meta:
        model  = User
        fields = ['username','email','password','phone_number',
                  'role','first_name','last_name']

    def clean(self):
        cleaned = super().clean()
        if cleaned.get("password") != cleaned.get("confirm_password"):
            raise forms.ValidationError("Passwords do not match")
        return cleaned

    def save(self, commit=True):
        user = super().save(commit=False)
        user.set_password(self.cleaned_data["password"])
        if commit:
            user.save()
            role = self.cleaned_data.get('role')
            if role == 'ORGANIZER':
                OrganizerProfile.objects.get_or_create(user=user)
            elif role == 'VENDOR':
                VendorProfile.objects.get_or_create(user=user)
            elif role == 'ATTENDEE':
                AttendeeProfile.objects.get_or_create(user=user)
        return user
```

Figure 75: *SignUpForm, unified registration for all roles*

6.2.6 signals.py and apps.py

signals.py acts as a safety net. Whenever a CustomUser is created by any path (the admin panel, the API, a fixture load, etc.), a post_save receiver automatically creates the matching profile row if it does not already exist. This keeps invariants that would otherwise have to be enforced at every write site. The receiver is registered in CoreConfig.ready() inside apps.py, which is the canonical Django pattern.

6.2.7 urls.py

core/urls.py is the routing table for the entire application. Routes are grouped by responsibility: authentication, role-specific dashboards, action endpoints for organizers and vendors, and the JSON APIs that power the dynamic dashboards. The AI assistant endpoint lives at the bottom of the file, exposed as /api/ai-assistant/. During development the file also serves media files (event banners and avatars) by appending the static() helper to urlpatterns when DEBUG is True.

6.2.8 views.py

views.py is the largest file in the project (over 1,100 lines) and contains every request handler. The file is organized into sections (general navigation and authentication, attendee views, business views, SCEGA admin views, JSON API endpoints, and the AI assistant) separated by comment banners. The discussion below covers the most representative responsibilities.

Authentication and access control:

Every protected view uses Django's @login_required decorator, and views that should be restricted to a specific role check request.user.role explicitly. The dashboard_view function acts as a central dispatcher: when an authenticated user lands on the home URL, it inspects their role and redirects them to the right dashboard.

```
# core/views.py

def dashboard_view(request):
    if request.user.is_authenticated:
        if request.user.role == 'ORGANIZER':
            return redirect('organizer_dashboard')
        elif request.user.role == 'VENDOR':
            return redirect('vendor_dashboard')
        elif request.user.role == 'SCEGA_ADMIN':
            return redirect('scega_dashboard')
        elif request.user.role == 'ATTENDEE':
            return redirect('attendee_dashboard')

    # Public landing: show APPROVED events
    events = Event.objects.filter(status='APPROVED').order_by('date')
    return render(request, 'core/index.html', {'events': events})
```

Figure 76: *Role-based dashboard dispatcher*

Page routing and form handling:

Each dashboard view collects exactly the data its template needs and passes it through Django's render() helper. The organizer_dashboard view illustrates how a single endpoint can handle multiple concerns: GETs render the dashboard, while POSTs branch on hidden form fields to either create an Event, edit an existing Event, or send a collaboration Request to a Vendor.

SCEGA licensing workflow:

One of Eventia's distinguishing features is the simulated SCEGA (Saudi Conventions & Exhibitions General Authority) licensing workflow. Every event an organizer creates enters the system with status='PENDING' and approval='PENDING'. SCEGA admins log into a separate dashboard (scega_dashboard) where they can approve or reject pending events. Only events with status='APPROVED' are visible to attendees and available for vendors to apply to.

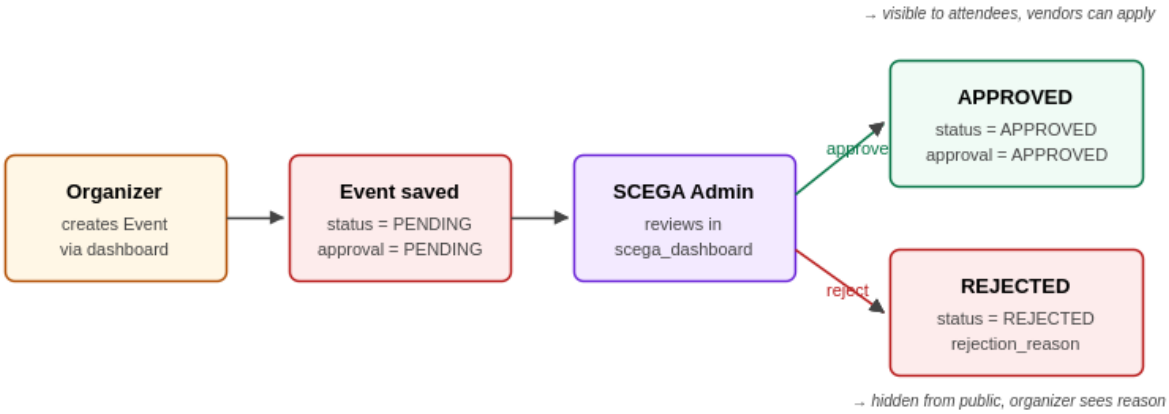

Figure 77: *SCEGA licensing workflow, from creation to publication*

JSON API endpoints:

All dashboard data flows through endpoints under /api/. We chose to write plain Django views that return JsonResponse rather than adopt Django REST Framework, because our API is consumed only by our own front-end and the data shape varies enough between roles that hand-written serialization was the simpler option. The api_organizer_analytics endpoint, for example, returns events, vendors, registrations, requests, messages, and broadcasts in a single payload, pre-aggregated so the front-end can build charts without round-tripping.

AI Assistant: Gemini integration:

The ai_assistant_chat endpoint is the most architecturally interesting view. It demonstrates careful resource management: the Gemini SDK is configured only once per process (rather than per request) via a module-level flag, the event-context string is cached for 5 minutes with automatic invalidation on Event.save(), and only the columns actually used in the prompt are pulled from the database via .only() and .select_related().

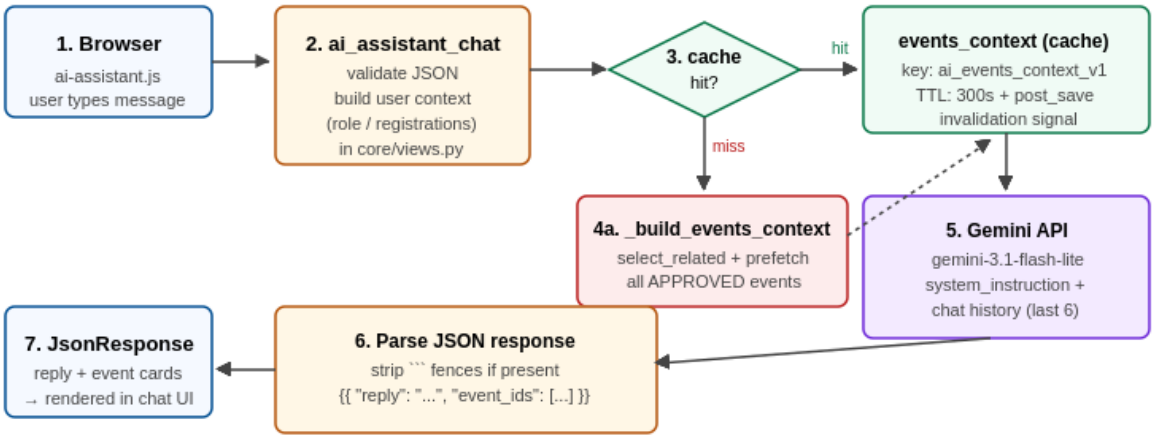

Figure 78: *AI assistant, request pipeline*

```
# core/views.py  (Gemini AI assistant, essentials)

_EVENTS_CACHE_KEY = 'ai_events_context_v1'
_EVENTS_CACHE_TTL = 300  # 5 minutes

@receiver(post_save, sender=Event)
def invalidate_events_cache(sender, **kwargs):
    """Clear the cache whenever ANY event is saved or edited."""
    cache.delete(_EVENTS_CACHE_KEY)

@csrf_exempt
def ai_assistant_chat(request):
    data = json.loads(request.body)
    user_message = data.get('message', '').strip()

    # Cached event context (5-min TTL, signal-invalidated)
    events_context = cache.get(_EVENTS_CACHE_KEY)
    if events_context is None:
        events_context = _build_events_context()
        cache.set(_EVENTS_CACHE_KEY, events_context, _EVENTS_CACHE_TTL)

    system_instruction = f"""You are the Eventia AI Assistant.
=== LIVE EVENT DATABASE ===
{events_context}
=== RULES ===
Output ONLY: {{"reply": "...", "event_ids": [int, ...]}}"""

    model = genai.GenerativeModel(
        'gemini-3.1-flash-lite',
        system_instruction=system_instruction,
        generation_config={"response_mime_type": "application/json"},
    )
    response = model.start_chat().send_message(user_message)
    return JsonResponse(json.loads(response.text))
```

Figure 79: *ai_assistant_chat, the Gemini pipeline*

6.2.9 admin.py

admin.py registers our models with Django's built-in admin interface, the auto-generated dashboard that lets superusers browse and edit data without touching the database directly. We provide customized ModelAdmin classes that control which columns are visible in list views, which fields can be filtered on, and which fields are searchable. Notably, the Event admin allows status and approval to be edited inline (via list_editable), giving SCEGA admins a fast bulk-approval workflow during testing.

6.2.10 migrations/

Django tracks schema evolution through migration files. As Eventia's data model evolved, each change to models.py produced a new migration: 0001_initial set up the original five tables, 0004 added rejection reasons and tightened gender choices, 0005 introduced withdrawal policies, 0006 added profile pictures to CustomUser, 0007 added job titles to AttendeeProfile, and 0008 added banner images to Event. Running python manage.py migrate replays this history deterministically, so a fresh database can be brought up to date in one command.

6.2.11 templates/

The templates/core/ directory contains all of the HTML rendered by Django views. Each role-specific dashboard is a large single-page application: organizer-dashboard.html (~2,000 lines) wires the create-event form, the event list, the vendor catalogue, and the analytics panel; attendee-dashboard.html (~900 lines) presents the browse-and-register interface; vendor-dashboard.html (~840 lines) shows incoming organizer requests and the in-app chat. Common header, footer, and authentication scaffolding live in shorter templates (login.html, signup.html, password-recovery.html). Every template is wrapped in {% load static %} and uses {% static 'css/...' %} tags to reference assets, includes {% csrf_token %} to satisfy Django's CSRF middleware, and renders dynamic content with {{ variable }} and {% for %} blocks.

6.2.12 static/

Static assets are organized by type. The css/ folder contains base.css for shared primitives and one stylesheet per role; the js/ folder mirrors that structure with one logic file per dashboard (attendee-logic.js, organizer-logic.js, organizer-analytics.js, vendor-logic.js, scega-logic.js) and one shared utility module (dashboard-ui.js). The ai-assistant.js file encapsulates the chat widget, including the floating sparkle button, suggested-chip rendering, and event-card injection.

6.2.13 tests.py

tests.py is the placeholder for our automated test suite. At the time of writing, end-to-end testing for the full workflow (signup, SCEGA approval, ticket purchase, vendor collaboration, broadcast) is performed manually using the development server and the Django admin to seed state. Chapter 7 (System Testing) covers the test scenarios in detail, and the structured test suite is being expanded to use Django's TestCase class and Client to automate those scenarios.

6.3 Summary

This chapter walked through the tools, frameworks, and code that bring Eventia to life. The architecture is intentionally conventional: a single Django application following the MVT pattern, MySQL for persistence, vanilla front-end technologies, and the standard PythonAnywhere deployment, which keeps the project understandable and maintainable. The non-conventional pieces are concentrated where they add the most value: a role-aware CustomUser with four roles, a hand-written JSON API tailored to each dashboard, a simulated SCEGA licensing workflow, and a Gemini-backed AI assistant with cache-friendly context generation. Together, these components implement the requirements set out in Chapter 4 and realise the design described in Chapter 5.

Chapter 7: System Testing

After completing the development and implementation phases, the Eventia platform was subjected to a structured system testing process to confirm its readiness for real-world use. While earlier testing layers, such as unit and integration testing, focus on isolated functions and the cooperation between individual modules, system testing examines the platform holistically, treating it as a single connected product rather than a collection of separate pieces.

The purpose of this phase is to detect issues that only emerge once every layer of the application operates together: the front-end interface, the Django back-end, the MySQL database, the JSON APIs, the role-based authentication flow, and the third-party integrations (Gemini for the AI assistant and Resend for transactional email). By executing realistic user journeys end-to-end, we verify that Eventia behaves correctly from the perspective of each of its four user roles: Attendee, Organizer, Vendor, and SCEGA Admin. In addition of platform guests who can browse events' dashboard.

The system testing carried out in this chapter targets the following objectives:

- **Confirming end-to-end workflows:** ensuring that complete user journeys, including account creation, event browsing, ticket purchase, event approval, vendor collaboration, and analytics review, run without breaking between stages.
- **Validating functional behaviour:** checking that each implemented feature, such as role-based dashboards, the SCEGA licensing workflow, the AI assistant recommendations, the in-app messaging, and the preparation-status tracker, produces the outcomes specified in earlier chapters.
- **Preserving data integrity:** confirming that records created by one role (e.g. a vendor request issued by an organizer) appear correctly for the other party, and that updates, withdrawals, and status changes leave the database in a consistent state.
- **Reviewing user experience:** confirming that pages render properly across the Arabic and English interfaces, that informative feedback is shown after every important action, and that navigation between dashboards is clear and predictable.

The remainder of this chapter presents a sequence of test scenarios that exercise the platform's most important paths. Each scenario follows a uniform structure, listing its identifier, objective, preconditions, executed steps, and the result observed against the expectation. Together, these scenarios provide a practical demonstration that Eventia satisfies its functional requirements and is prepared for deployment.

7.1 Test Scenarios:

7.1.1 Test Scenario: New User Signup and Login

Test ID: ST-001

Scenario Title: Successful User Registration and Login

Objective: Verify that a new user can successfully register an account and subsequently log in to access the application's home page.

Preconditions:
- The application is running.
- The user is not currently logged in.
- The desired username and email are not already taken.

Test Steps:
1. Navigate to the application's landing page (/).

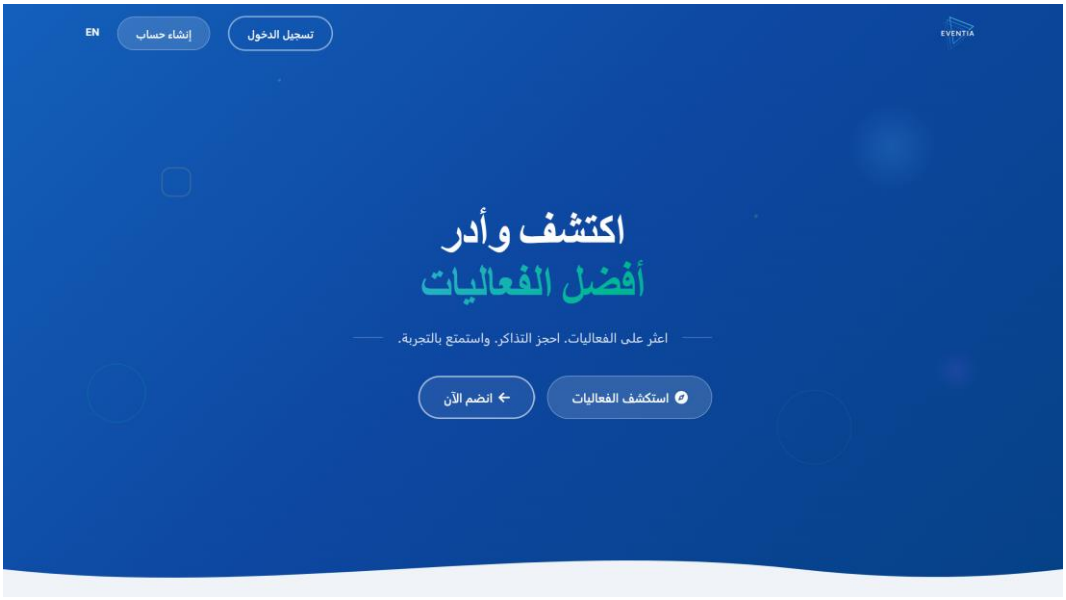

Figure 80: *Landing page with "Signup" link*

2. Click the "Signup" or "إنشاء حساب".
3. Verify the signup page (/signup/) loads, displaying the Signup Form.

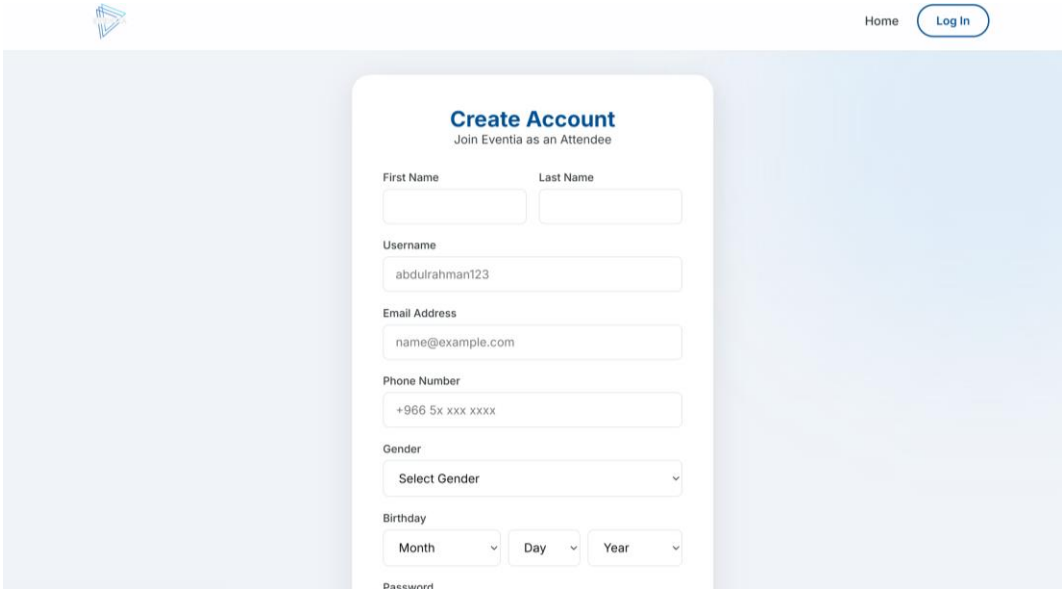

Figure 81: *Signup form displayed on /signup/ page*

4. Choose the role then fill in all required fields, for an attendee: Username, First Name, Last Name, Email, Phone Number, Gender, Birthday, Password, Confirm Password.
5. Click the "انشاء الحساب" button.
6. The system shall automatically logged in using the registered account and redirect to specific dashboard.

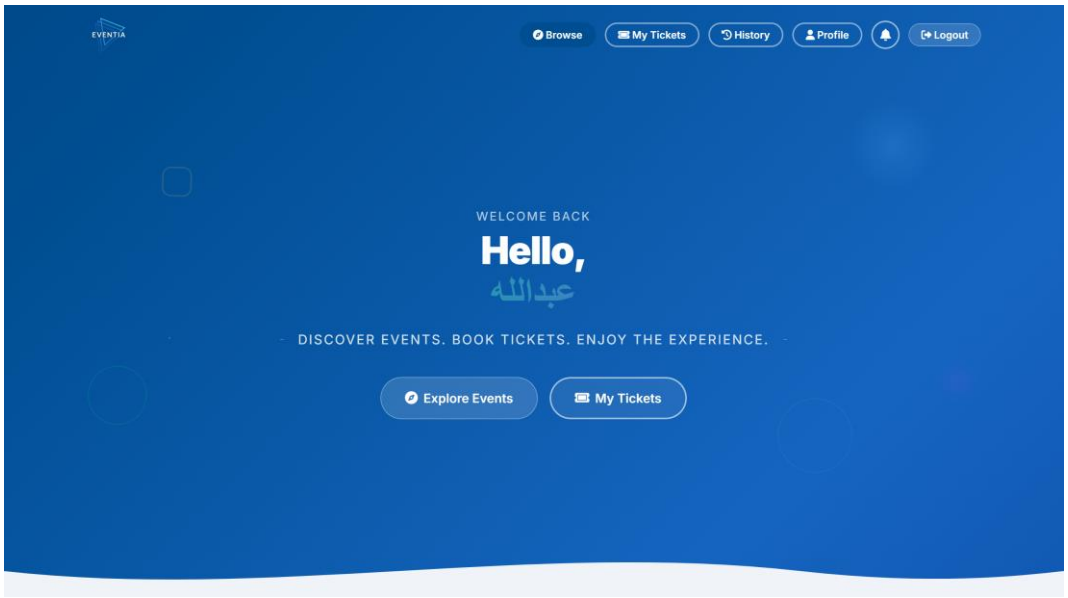

Figure 82: *Registration success and the system logged in*

7. For the login process, Enter the username and password used during registration.
8. Click the "Login" button.

9. Observe the system response (redirection to home page).

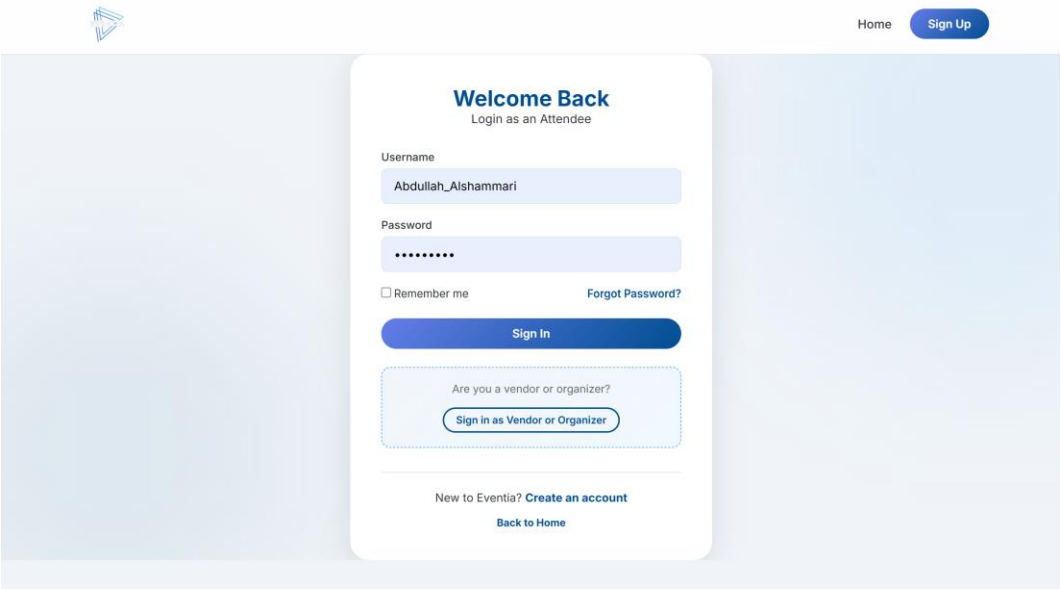

Figure 83: *login form accepts the same provided user credentials*

Expected Result:
• After step 5, the user should be redirected to the main dashboard with a new user record created in the database.
• After step 9, the user should be redirected to the main dashboard (/dashboard/).

Match Actual Result so Status: Pass Test ST-001

7.1.2 Test Scenario: Guest/Attendee Browsing events with Searching and Filtering

Test ID: ST-002

Scenario Title: Successful event Browsing and using searching and filtering features for Attendees and Guests

Objective: Verify that an attendee/guest can successfully browse list of existing events, check a specific event details, filter events using search of categories.

Preconditions:
• The application is running.
• There exist some active events.
• The user is not currently logged in as Organizer or Vendor.

Test Steps:
1. Navigate to the application's landing page.
2. Scroll down or click "Explore events" / "استكشف الفعاليات" button.

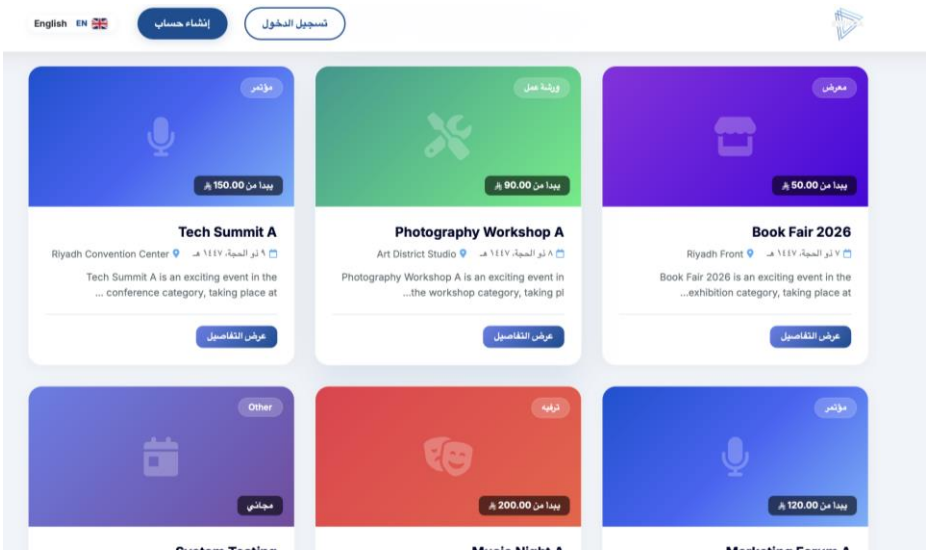

Figure 84: *The platform shows list of existing events*

3. Type in the search bar your interests or use categories filters.

4. Click “View details” or “عرض التفاصيل” for checking important details such as date, capacity.

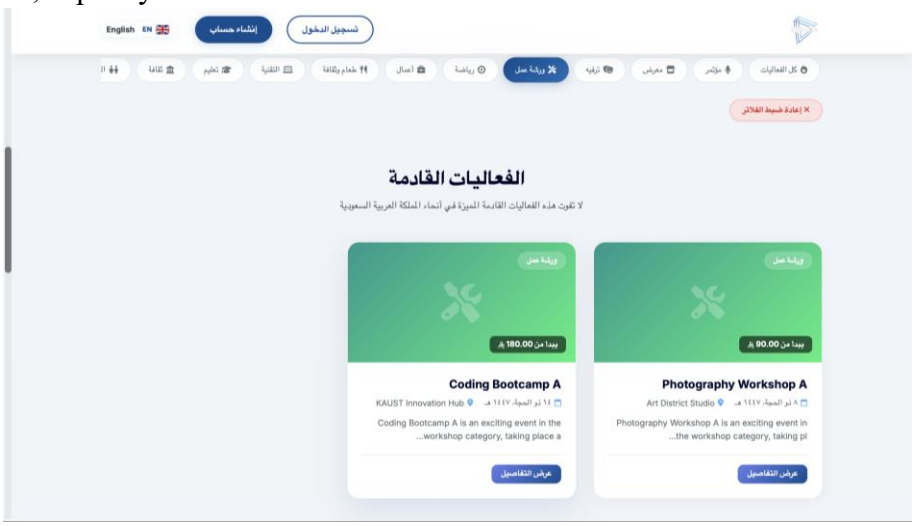

Figure 85: *Searching and Filtering events features*

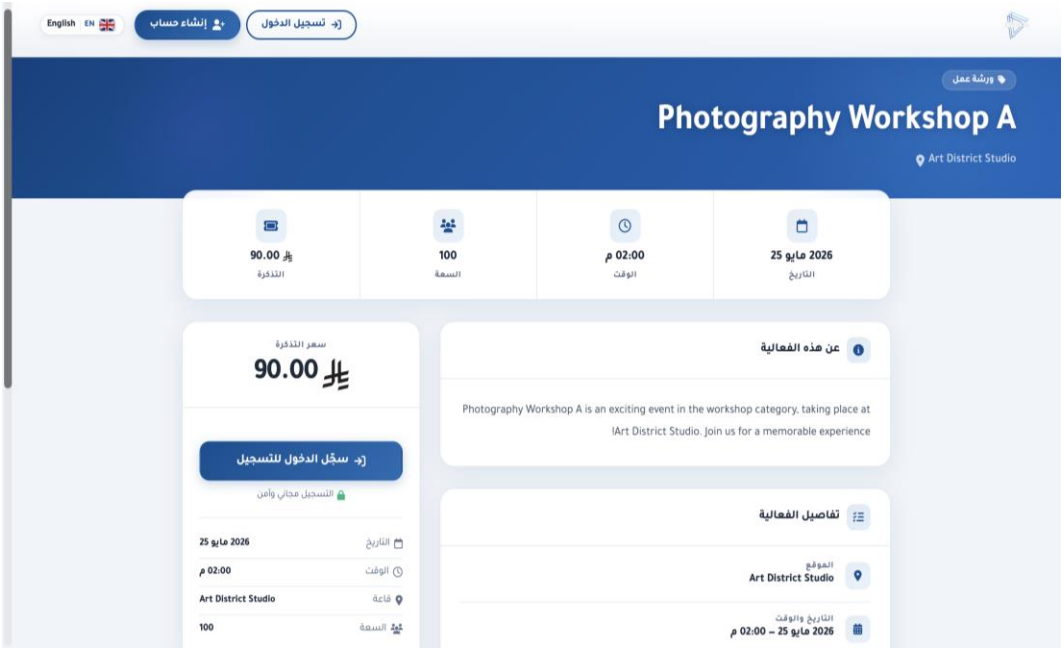

Figure 86: *Event details functionality*

Expected Result:
- The system shows list of all active events.
- After filtering the system shows all compliant events.
- All related details show to the user when clicking view details.

Match Actual Result so Status: Pass Test ST-002

7.1.3 Test Scenario: Attendee register a ticket for a specific event

Test ID: ST-003

Scenario Title: Successful event registration for attendees.

Objective: Verify that an attendee can successfully register for an active event, pay and get the ticket.

Preconditions:
- The application is running.
- The event's tickets is not sold out.
- The user is logged in as an Attendee.

Test Steps:
1. Navigate to the application's landing page.
2. Reach to the specific event via Browsing, Searching or Filtering.
3. Click "Register" on the event card.

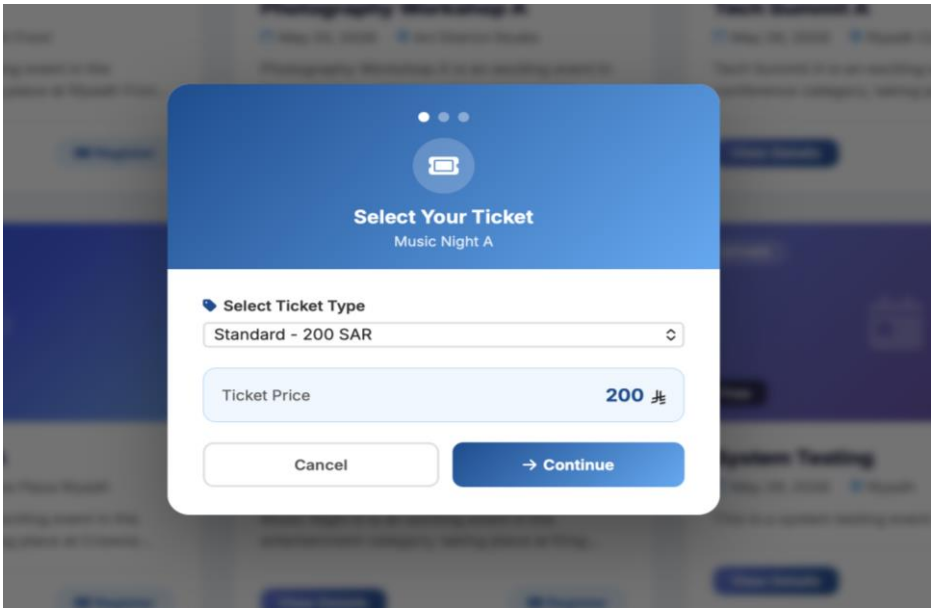

Figure 87: *Event's Registration - Ticket type Selection*

4. Choose the ticket type then click “Continue” for payment.

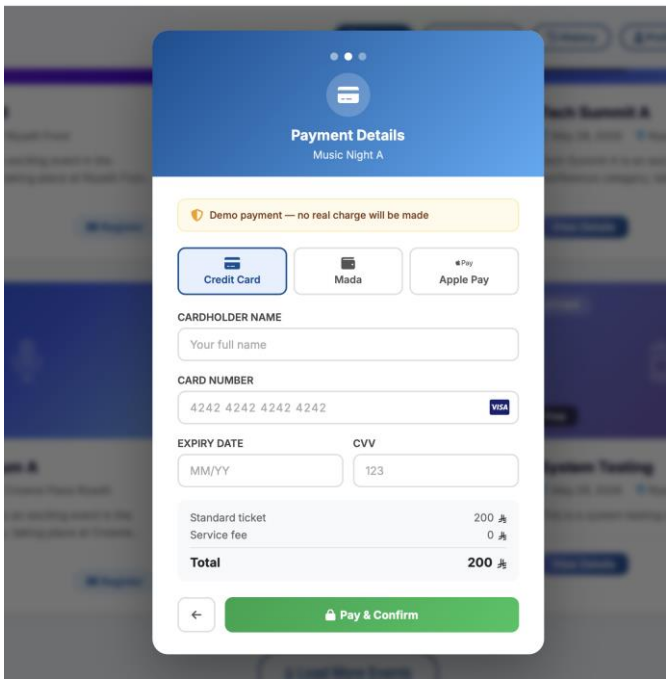

Figure 88: *Event's Registration - Ticket type Selection*

5. Fill in payment Information then click “Pay & Confirm” to receive the ticket.

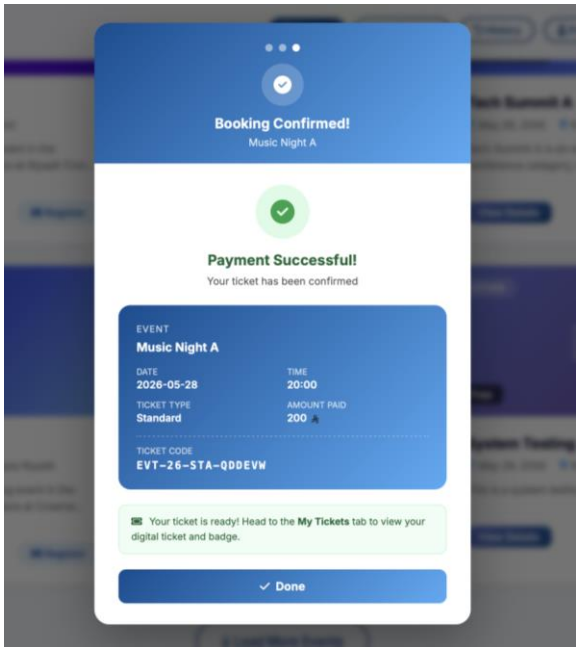

Figure 89: *Ticket Registration Success*

Expected Result:
- Ticket issues to the Attendee in the system.
- Attendee receive the ticket in “My tickets” section.
- Attendee is able to print, show or save the ticket in the email.

Match Actual Result so Status: Pass Test ST-003

7.1.4 Test Scenario: Organizer Event Creation

Test ID: ST-004

Scenario Title: Successful event creation for organizers.

Objective: Verify that an organizer can successfully create an event and request approval from SCEGA for publishing.

Preconditions:
- The application is running.
- The user is logged in as an Organizer.

Test Steps:
1. Navigate to the Organizer dashboard (Automatically after logging in).
2. Click “Create Event” or "إنشاء فعالية" tab from the navigation bar.

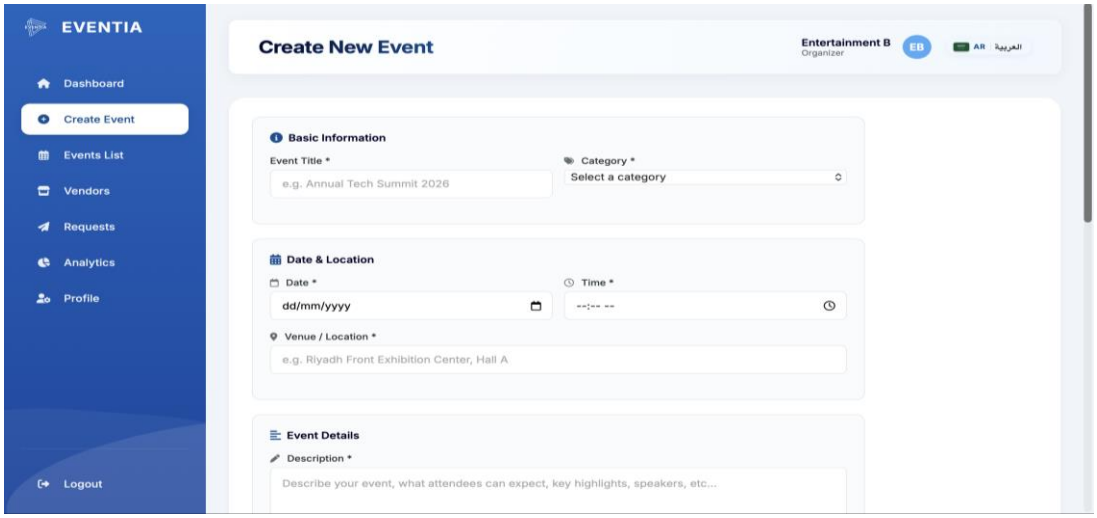

Figure 90: *Organizer dashboard*

3. Fill in details : Basic Information, Data & Location, Event Details, Withdrawal policies and Tickets.
4. Upload Event banner (if any).
5. Submit, The event shall be at “Pending” state for approval, the state can be checked via Events List tab at the same navigation bar until SCEGA’s admin accepts it.

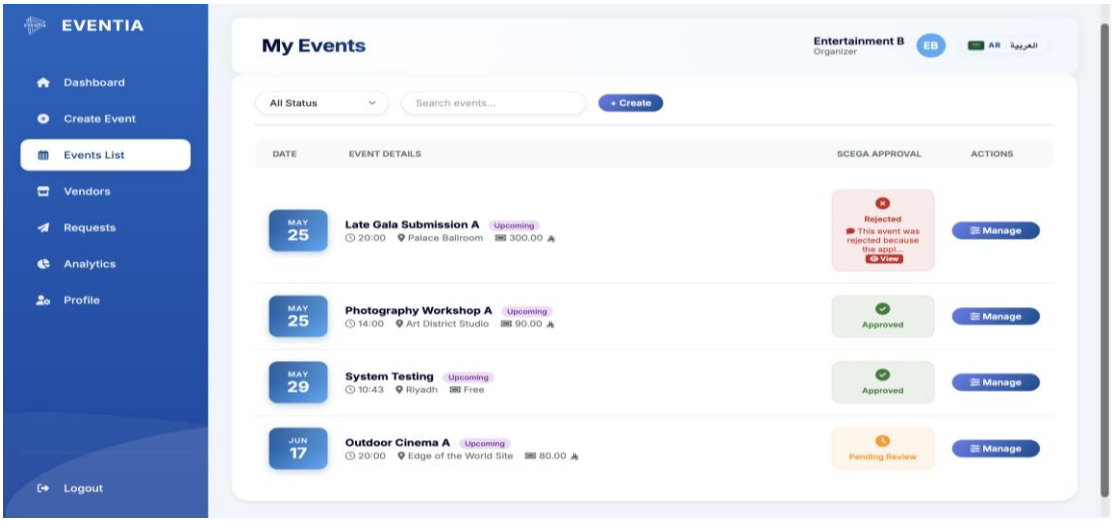

Figure 91: *Event state after submission*

Expected Result:

• The event is saved in the database as Pending state.
• Organizer can check events states anytime at Events List tab.
• The event is shown at the landing page for Attendees/Guests when the event is approved by SCEGA’s admin.

Match Actual Result so Status: Pass Test ST-004

7.1.5 Test Scenario: SCEGA Admin Reviews and Approves a Pending Event

Test ID: ST-005

Scenario Title: Successful event review and approval by SCEGA Admin

Objective: Verify that a SCEGA Admin can log into the SCEGA dashboard, review a pending event submitted by an organizer, and approve it so that the event becomes visible to attendees and open to vendor applications.

Preconditions:

- The application is running.
- At least one event with status "Pending" exists in the system (e.g. the event created in ST-004).
- A SCEGA Admin account is registered and the user is not currently logged in.

Test Steps:

1. Navigate to the SCEGA login page (/scega/).

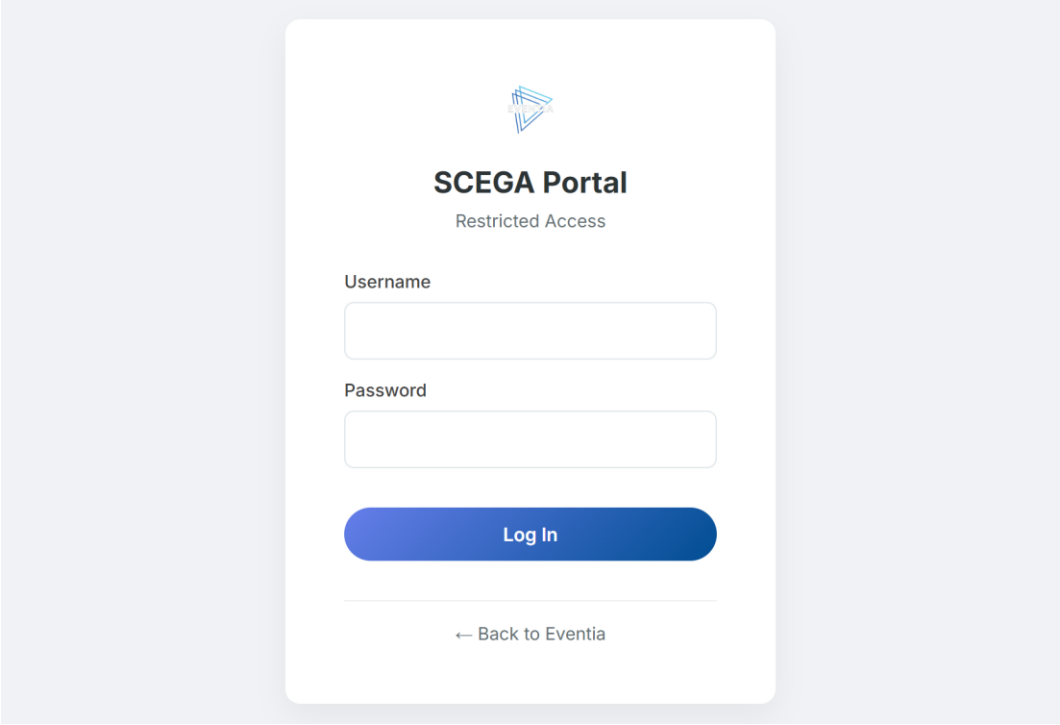

Figure 92: *SCEGA Admin login page*

2. Enter the SCEGA Admin credentials and click the "Login" button.

3. The system automatically redirects to the SCEGA dashboard (/dashboard/scega/).

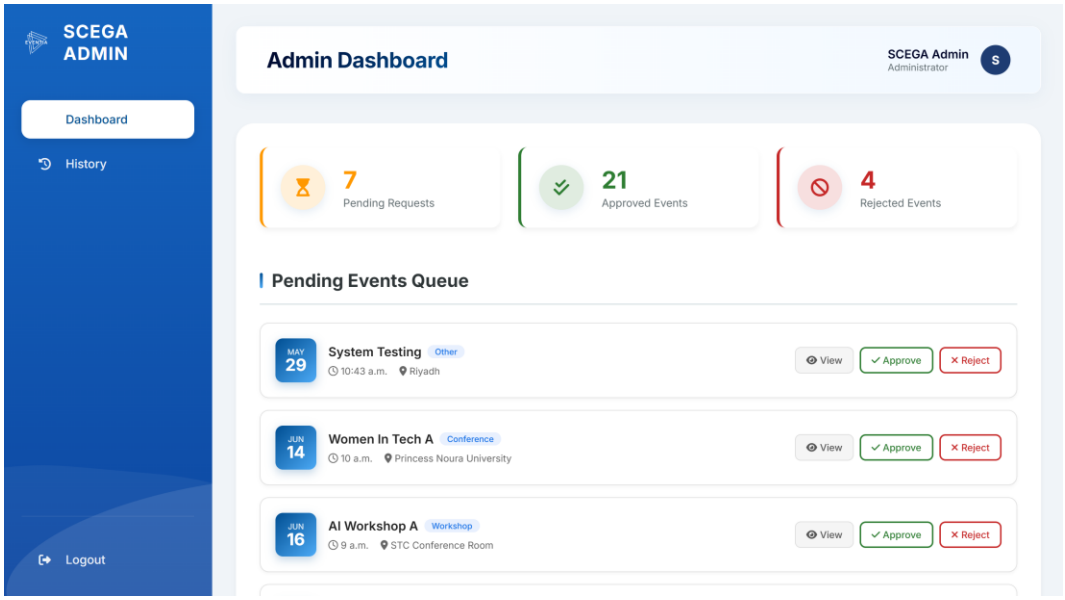

Figure 93: *SCEGA Admin dashboard showing pending events list*

4. Locate the pending event in the list and click "View Details" to review the event information (organizer, title, date, location, capacity, ticket price, age restriction, banner).

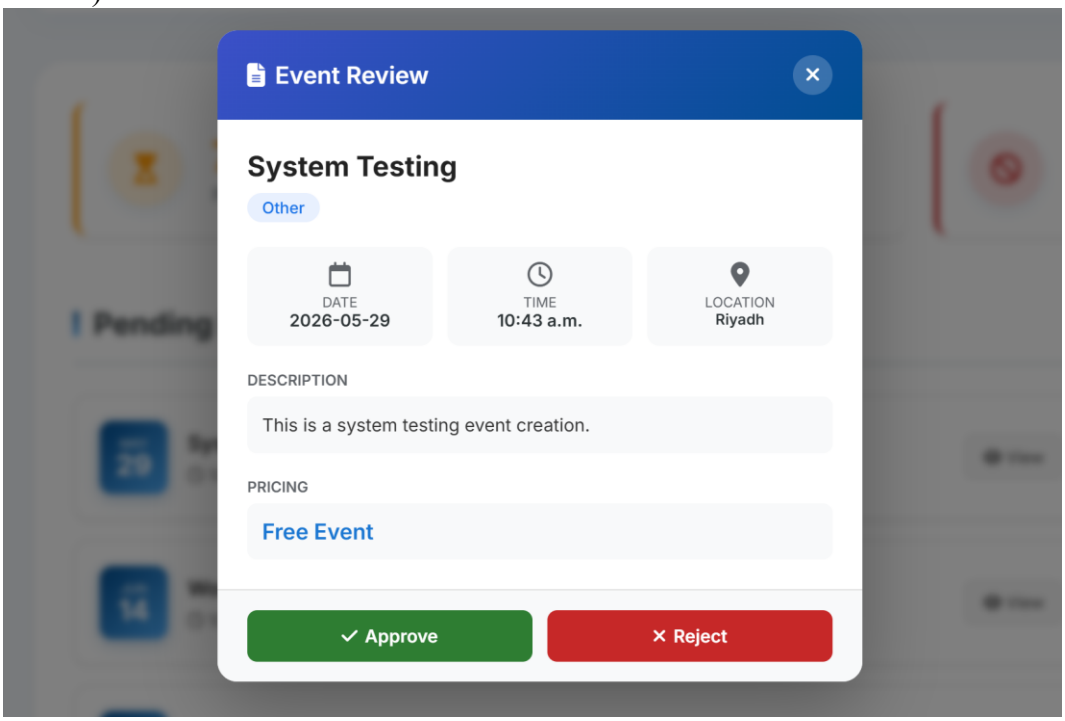

Figure 94: *Event details view inside SCEGA dashboard*

5. Click the "Approve" or "موافقة" button to issue the license.

6. Observe the system response and the updated event status.

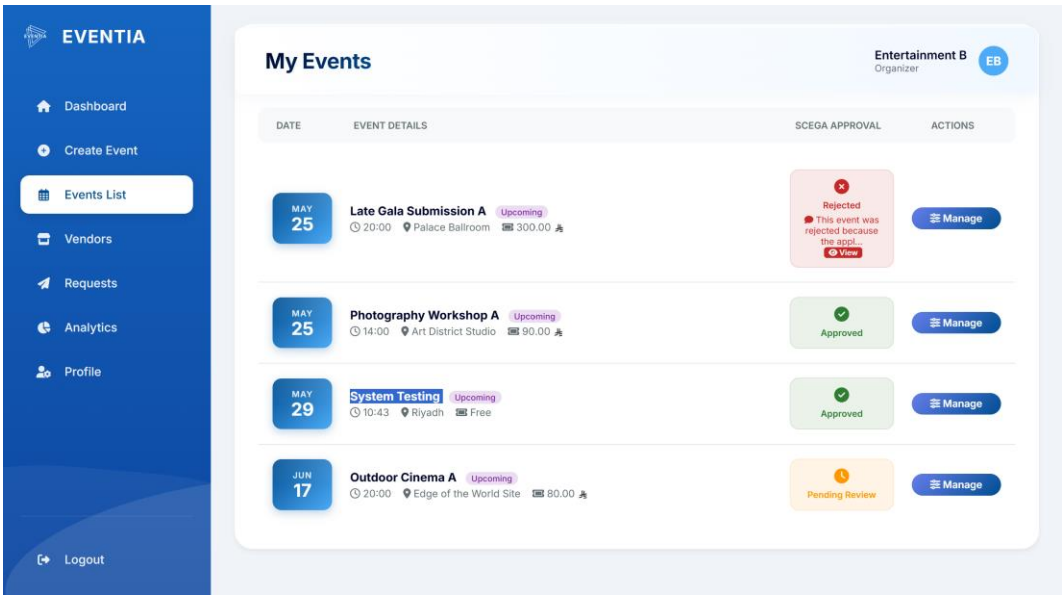

Figure 95: *Event status updated to "Approved" after SCEGA decision*

Expected Result:

- The event's status field is updated from "Pending" to "Approved" in the database.
- The event disappears from the SCEGA pending list and appears in the approved list.
- The approved event becomes publicly visible on the landing page for attendees and guests.
- Vendors can now apply to collaborate on the event from their dashboard.
- The organizer sees the updated status reflected in their "Events List" tab.

Match Actual Result so Status: Pass Test ST-005

7.1.6 Test Scenario: Organizer Sends a Collaboration Request to a Vendor

Test ID: ST-006

Scenario Title: Successful vendor collaboration request from Organizer

Objective: Verify that an organizer can browse the vendor catalogue, select a vendor for a specific approved event, send a collaboration request with a custom message, and confirm the request reaches the vendor's incoming requests panel.

Preconditions:

- The application is running.
- The organizer owns at least one approved event.
- At least one vendor profile is registered in the system.

- The user is logged in as an Organizer.

Test Steps:

1. Navigate to the Organizer dashboard (Automatically after logging in).
2. Click "Vendors" or "الموردون" tab from the navigation bar to open the vendor catalogue.

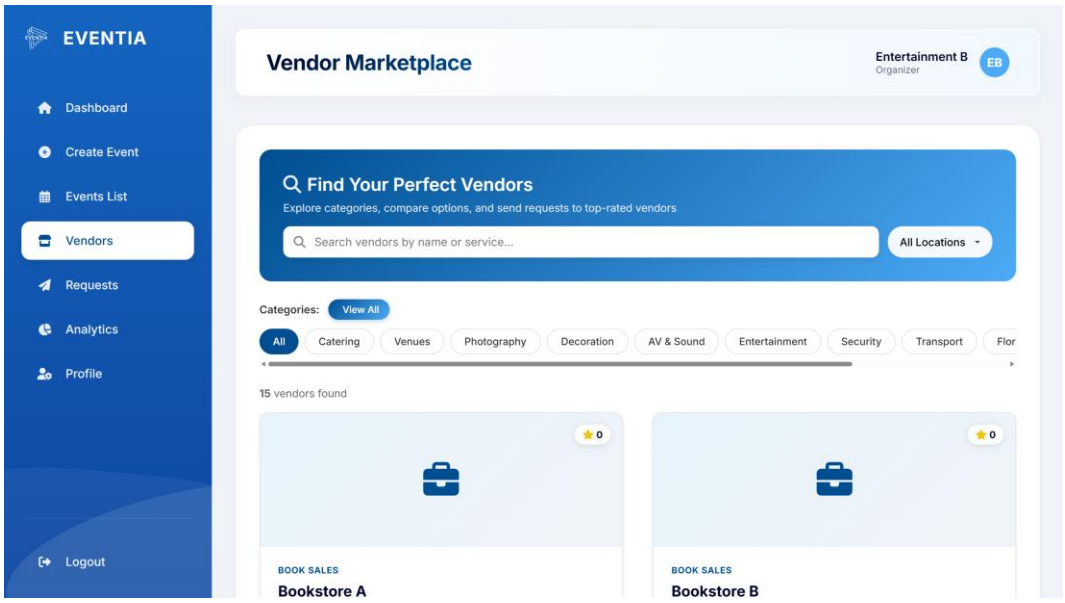

Figure 96: *Vendor catalogue inside Organizer dashboard*

3. Browse the available vendors filtered by service type and click "Send Request" or "إرسال طلب" on the chosen vendor card.
4. In the request modal, select the target event from the dropdown of the organizer's approved events.

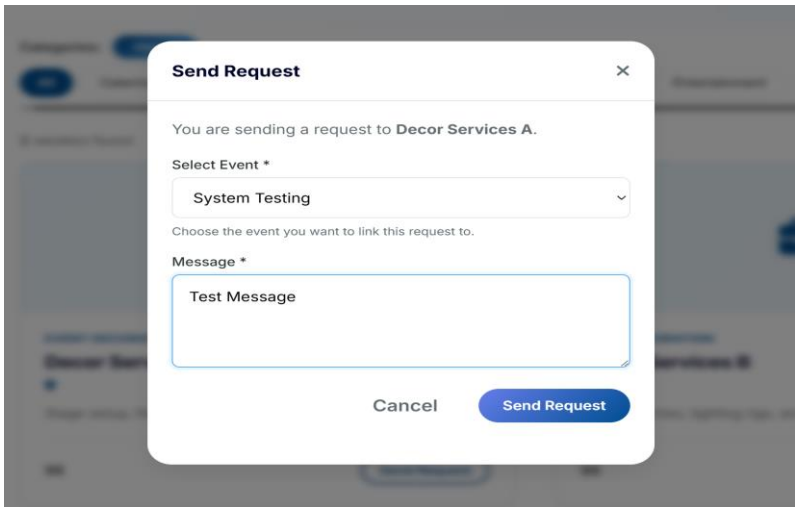

Figure 97: *Send Request modal with event selection and message field*

5. Write a short message describing what is needed (e.g. "We need catering for 200 attendees").
6. Click "Submit Request" or "إرسال" to dispatch the request.
7. Observe the success confirmation and the new entry in the "Outgoing Requests" section.

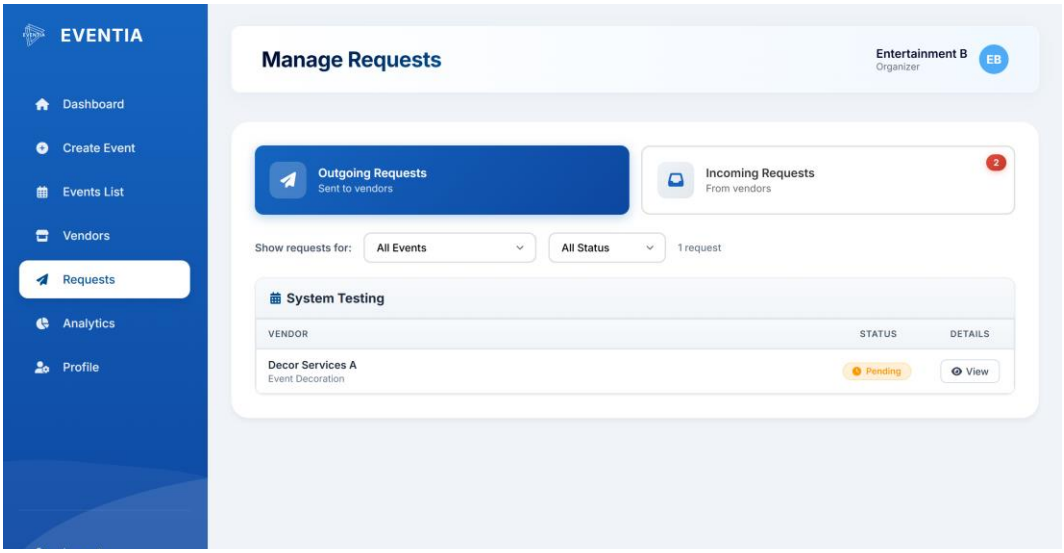

Figure 98: *Outgoing request listed with "Pending" status*

Expected Result:

- A new Request record is saved in the database with status "Pending" and sent_by = "ORGANIZER".
- The request appears in the organizer's outgoing requests list with the correct event and vendor.
- The vendor sees the new request in their incoming requests panel after login.
- Duplicate requests for the same event/vendor pair are prevented by the system with an appropriate warning message.

Match Actual Result so Status: Pass Test ST-006

7.1.7 Test Scenario: Vendor Accepts a Request and Updates Preparation Status

Test ID: ST-007

Scenario Title: Successful vendor acceptance and preparation status tracking

Objective: Verify that a vendor can review an incoming collaboration request, accept it, and progressively update the preparation status through the defined lifecycle (Pending, Preparing, In Transit, Setting Up, Ready) so that the organizer is kept informed.

Preconditions:

- The application is running.

- At least one pending request addressed to this vendor exists (e.g. the request sent in ST-006).
- The user is logged in as a Vendor.

Test Steps:

1. Navigate to the Vendor dashboard (Automatically after logging in).
2. Open the "Incoming Requests" or "الطلبات الواردة" tab from the navigation bar.

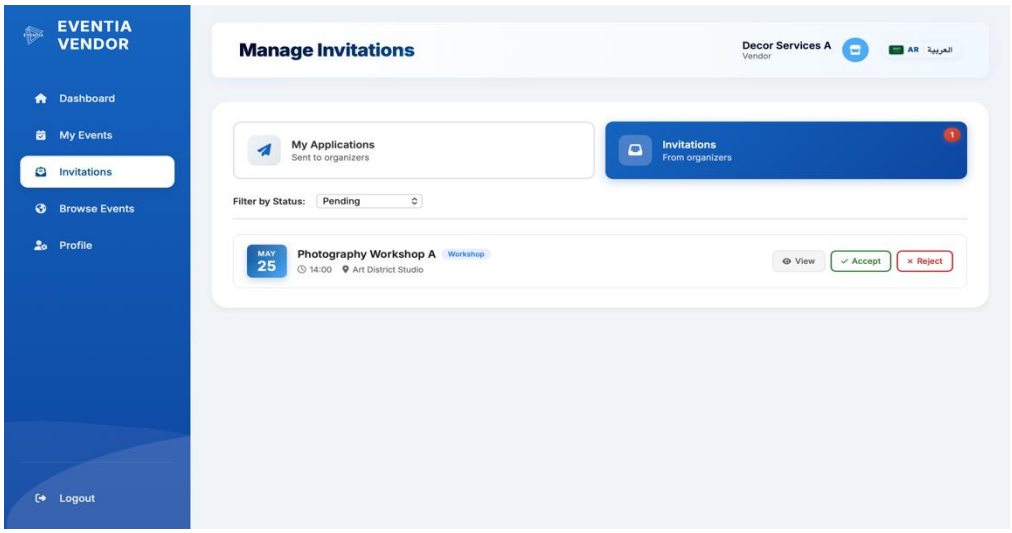

Figure 99: *Vendor dashboard with incoming requests list*

3. Click on the pending request to view its details, including the event information and the organizer's message.
4. Click "Accept" or "قبول" to confirm participation in the event.

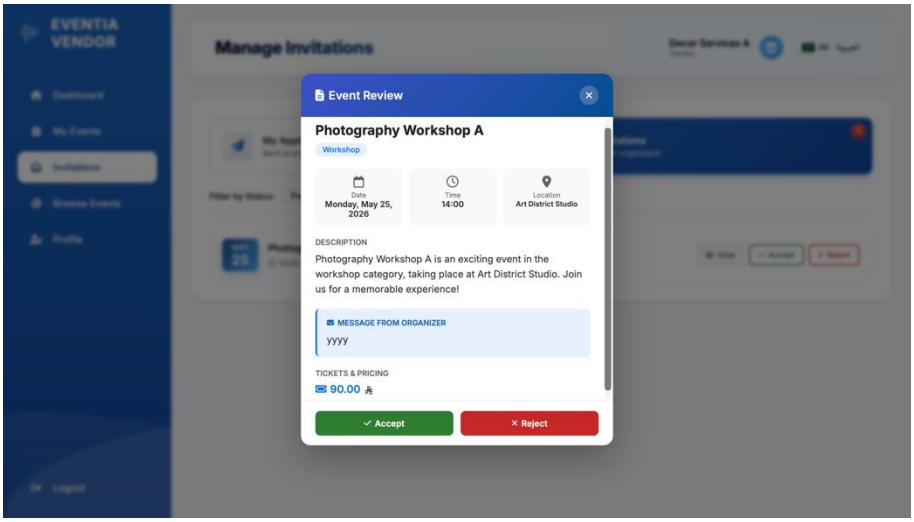

Figure 100: *Request detalis panel with event and message*

5. Observe that the request status changes to "Approved" and that the preparation status tracker becomes available.
6. Update the preparation status to "Preparing" using the dropdown control.

7. Later, change the preparation status to "In Transit", then "Setting Up", and finally "Ready" as the work progresses.

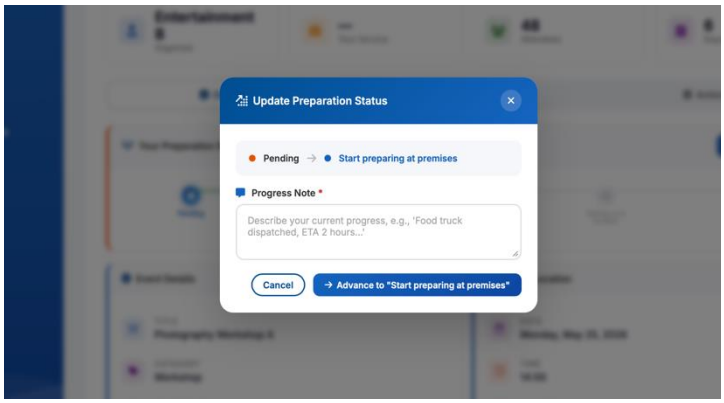

Figure 101: *Vendor updates prepration status for a specific event*

Expected Result:

- The Request record's status field changes to "Approved" in the database.
- A new VendorStatusUpdate record is created for every preparation-status change, building a complete audit trail.
- The organizer sees the live preparation status of each vendor on their event detail page.
- The vendor remains listed as an approved vendor on the event throughout the lifecycle.

Match Actual Result so Status: Pass Test ST-007

7.1.8 Test Scenario: Organizer and Vendor Communicate via In-App Messaging

Test ID: ST-008

Scenario Title: Successful bidirectional in-app messaging between Organizer and Vendor

Objective: Verify that the organizer and the vendor can exchange messages inside the platform for a specific event, that messages persist across sessions, and that the recipient sees new messages without leaving their dashboard.

Preconditions:

- The application is running.
- There exists an approved event with at least one accepted vendor (e.g. the outcome of ST-007).
- Both the organizer and the vendor have active accounts.

Test Steps:

1. Log in as the Organizer and open the relevant event from "Events List".
2. Go to the Communication Tab then click the required vendor to open the chat panel.

3. Type a message (e.g. "Please arrive an hour before the event starts") and click the send button.
4. Observe the message appear immediately in the chat thread.

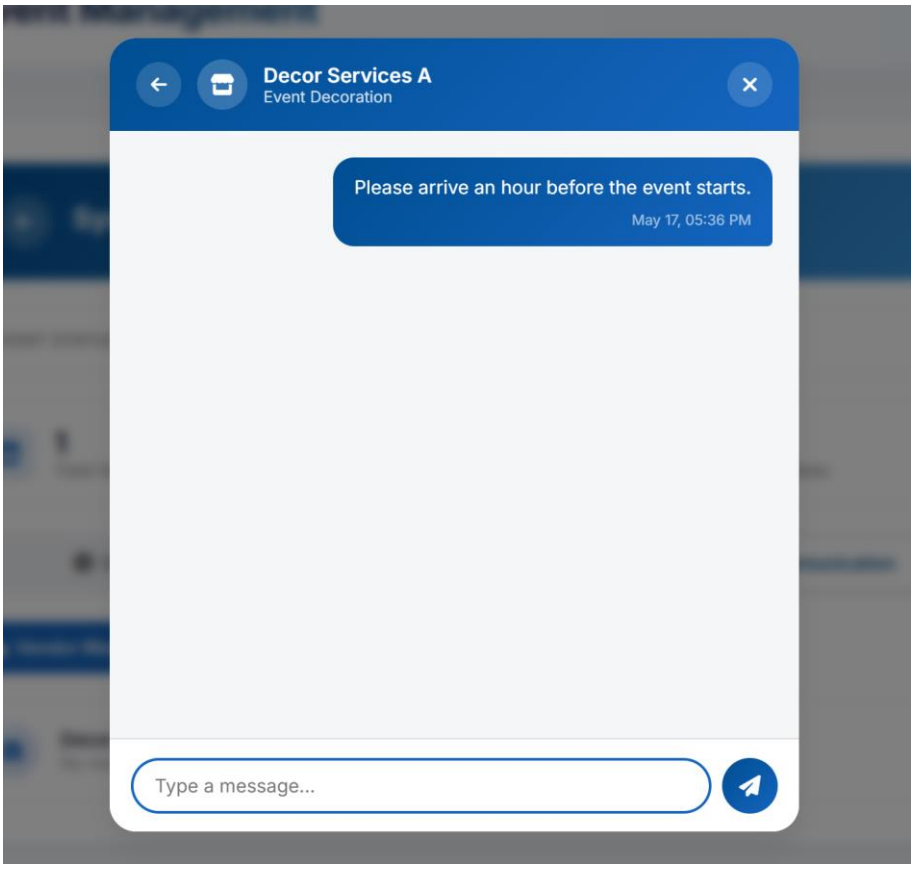

Figure 102: *Organizer-side chat panel for a specific vendor*

5. Log out and then log in with the vendor account.
6. For the vendor side, Navigate to the "Messages" or "الرسائل" section of the vendor dashboard.
7. Type a reply (e.g. "Confirmed, we will be on site by 5 PM") and click send.

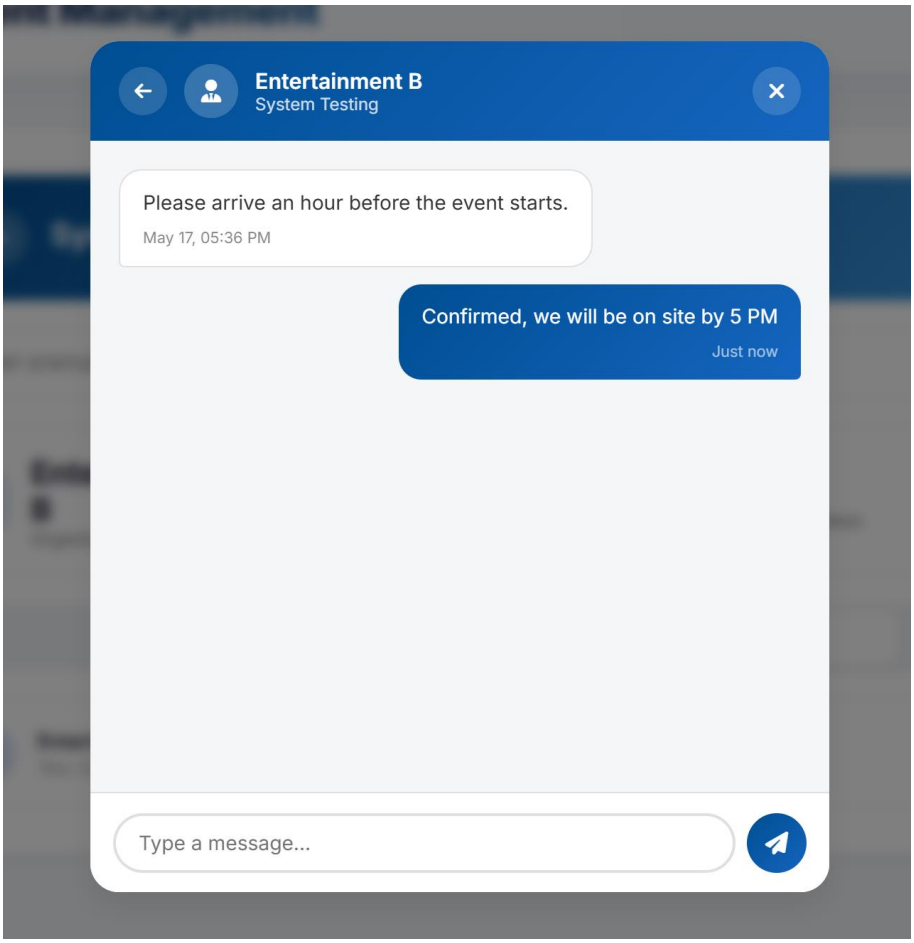

Figure 103:*Updated chat thread with both messages*

8. Log back in as the Organizer and verify that the vendor's reply is visible in the same chat thread.

Expected Result:

- Each sent message is persisted as a Message record in the database with the correct sender, event, vendor, text, and timestamp.
- Messages appear chronologically inside the chat panel for both parties.
- The chat history is preserved across logout and login cycles.
- Messages are scoped to the relevant event so that conversations from other events do not interfere.

Match Actual Result so Status: Pass Test ST-008

7.1.9 Test Scenario: AI Assistant Recommends Events to Attendees and Guests

Test ID: ST-009

Scenario Title: Successful event recommendation through the Gemini-powered AI Assistant

Objective: Verify that any visitor on the landing page can interact with the AI assistant, receive a natural-language reply, and see clickable event cards generated from approved events in the database.

Preconditions:

- The application is running.
- At least three events have been approved and are publicly visible.
- The Gemini API key is configured and the service is reachable.

Test Steps:

1. Navigate to the application's landing page (/) as a guest or logged-in attendee.
2. Click the floating AI assistant button near the search bar.

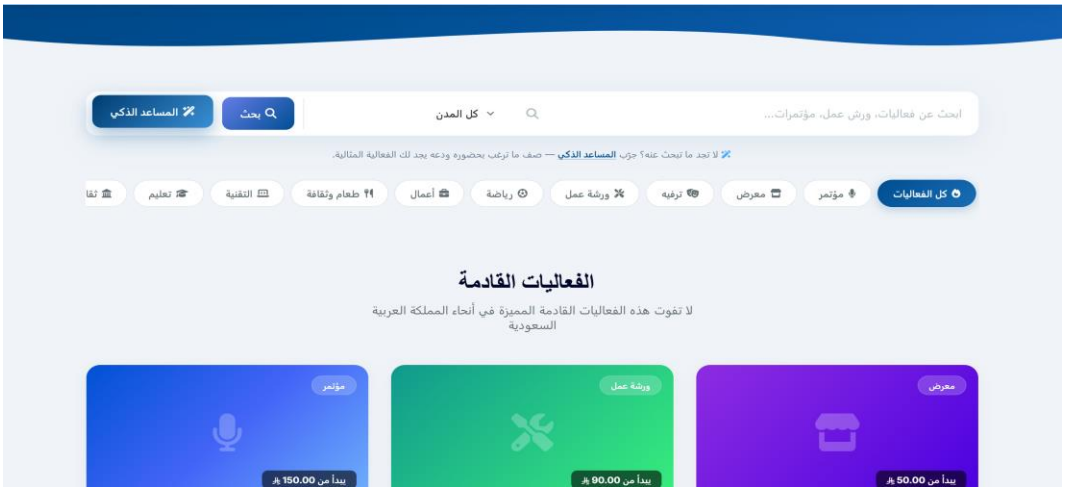

Figure 104: *AI Assistant button and opened chat window*

3. Click one of the suggested chips (e.g. "What events are happening this weekend?") or type a custom question in the input field.

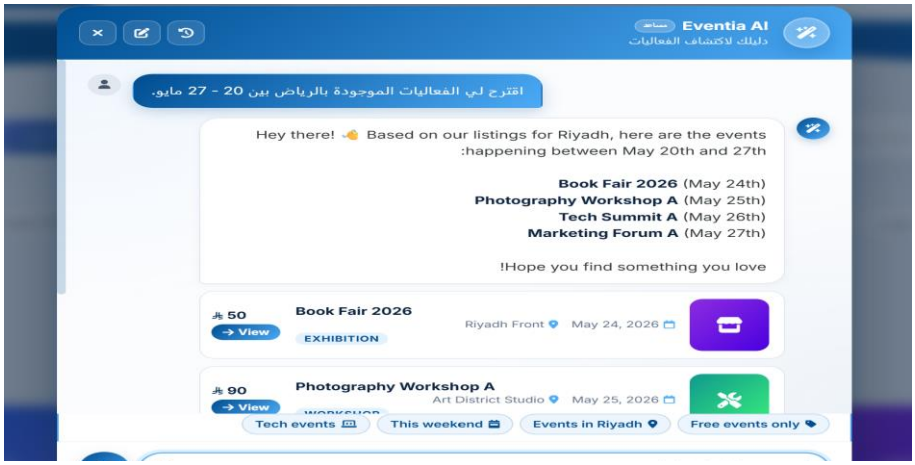

Figure 105: *AI Assistant responding with text and event cards*

4. Wait for the assistant's reply and observe the rendered event cards.
5. Click on one of the event cards to navigate to the corresponding event details page.
6. Return to the assistant and ask a follow-up question (e.g. "Are any of them free?") to confirm conversation context is maintained.

Expected Result:

- The assistant returns a coherent reply that respects the user's question.
- Returned event cards correspond to actual approved events in the database (no fabrications).
- Each event card is clickable and navigates correctly to the event details page.
- The conversation history is preserved within the session, enabling natural follow-ups.
- When the same query is asked again quickly, the response uses the cached event context for faster turnaround.

Match Actual Result so Status: Pass Test ST-009

7.1.10 Test Scenario: Organizer Reviews Analytics Dashboard

Test ID: ST-010

Scenario Title: Successful access and exploration of analytics charts by the Organizer

Objective: Verify that an organizer can open the analytics dashboard, view multiple chart types reflecting actual platform data (registrations, revenue, vendor coverage, attendee demographics), and filter the analytics by a specific event.

Preconditions:

- The application is running.
- The organizer owns at least one approved event with one or more ticket sales.
- The user is logged in as an Organizer.

Test Steps:

1. Navigate to the Organizer dashboard (Automatically after logging in).
2. Click the "Analytics" or "التحليلات" tab from the navigation bar.

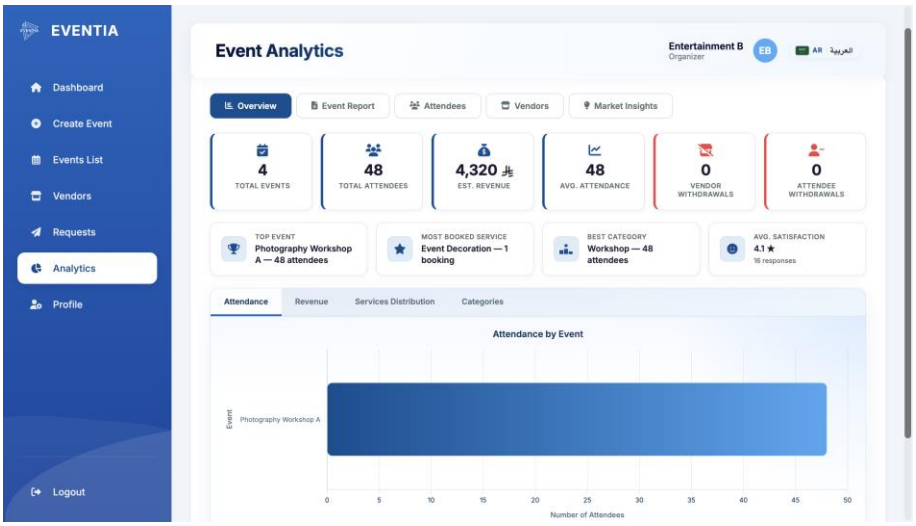

Figure 106: *Analytics dashboard with summary KPIs at the top*

3. Review the top-level KPIs: total events, total registrations, total revenue, and approved vendors.
4. Scroll to view each chart in turn: registrations over time (line chart), revenue per event (bar chart), attendee gender split (doughnut chart), and vendor coverage (area chart).
5. Use the event selector dropdown to filter the analytics view to a single event.
6. Observe that all charts and KPIs update to reflect the chosen event only.
7. Switch between tabs (e.g. "Engagement", "Revenue", "Vendors") to confirm each panel renders without errors.

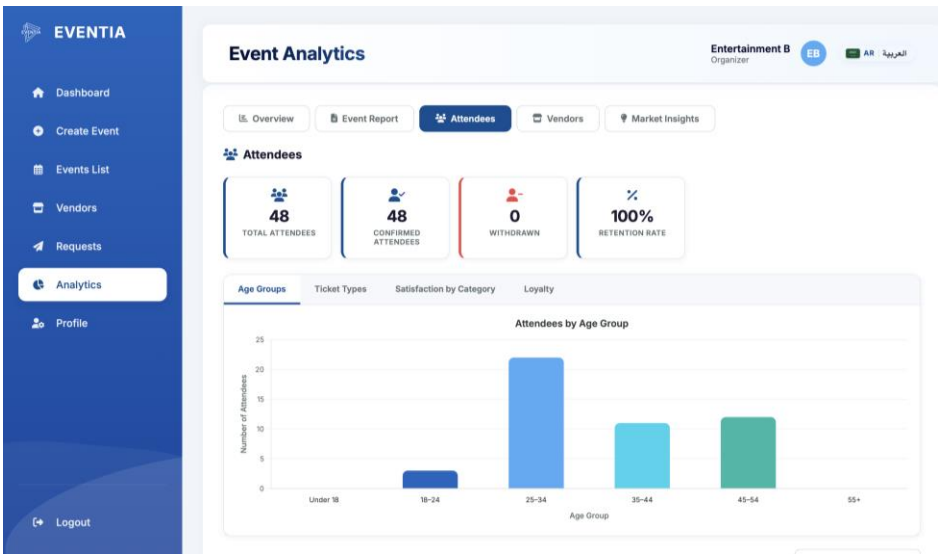

Figure 107: *Charts Panel for Events Analytics*

Expected Result:

- The analytics endpoint (/api/organizer/analytics/) returns a successful JSON payload.
- All charts render correctly and the data shown matches the underlying database (events, tickets, requests, messages, broadcasts).
- Filtering by event narrows the dataset across every visible chart and KPI simultaneously.
- Switching between analytics tabs does not break the page or duplicate chart instances.
- The analytics page remains responsive even with multiple events and large numbers of tickets.

Match Actual Result so Status: Pass Test ST-010

Chapter 8: Conclusion

In conclusion, this project successfully created Eventia, a central platform designed to modernize and improve how events are managed in the Saudi Arabian market. By bringing traditional, scattered processes into a single digital system, the platform connects event organizers, government authorities, logistics vendors, and attendees in one place. The main value of Eventia is its ability to help these different groups work together smoothly, ensure that events follow official regulations, and provide useful data insights to make event planning faster and more organized.

Looking back at what the project achieved, several key milestones stand out. First, the platform makes official approvals much easier by building local regulatory rules directly into the event creation process, which helps avoid administrative delays before an event is published. Second, Eventia improves how organizers and external vendors work together by moving invitations, negotiations, and approvals online, which removes communication delays and makes teamwork clearer. To make the system easy to use, the platform offers specific dashboards tailored to each user's actual needs: organizers can oversee the entire timeline, vendors can manage their services easily, and attendees enjoy a smooth experience from finding an event to buying a ticket. Additionally, the platform acts as an intelligent partner rather than just a basic database; by analyzing platform data and system analytics, it gives organizers clear insights into ticket sales, revenue, and audience interest to help them make better business decisions. Finally, the platform completely respects local culture and consumer preferences, creating a trusted and highly accessible experience for everyone in the local market.

Building upon these foundational achievements, future work will focus on expanding Eventia’s capabilities through advanced features and moving the project into a commercial reality. First, the platform will introduce a personalized recommendation engine that analyzes user history and preferences to suggest relevant events to attendees. Second, vendor operations will be enhanced by integrating their point-of-sale systems, providing them with valuable data analytics to track sales performance and inventory trends directly through the platform. Third, real-time tracking will be introduced by connecting the platform to smart gates and sensors, allowing organizers to monitor crowd density and live check-in or check-out rates for better crowd management and safety. Fourth, the system will incorporate automated topic mining; instead of organizing events manually, machine learning algorithms will automatically cluster and categorize events based on their descriptions and user discussions. Ultimately, the next phase will also focus on a strategic marketing campaign to launch Eventia in the commercial market, transforming this project from an academic prototype into a real-world enterprise.

References

[1] GeeksForGeeks, Feb 2025. [Online]. Available: https://www.geeksforgeeks.org/blogs/front-end-development/.

[2] W3Schools, Sep 2025. [Online]. Available: https://www.w3schools.com/html/html_intro.asp.

[3] W3Schools, Sep 2025. [Online]. Available: https://www.w3schools.com/css/css_intro.asp.

[4] Javascript.info, Sep 2025. [Online]. Available: https://javascript.info/intro.

[5] GeeksForGeeks, Sep 2025. [Online]. Available: https://www.geeksforgeeks.org/bootstrap/bootstrap-tutorial/.

[6] TutorialsPoint, Sep 2025. [Online]. Available: https://www.tutorialspoint.com/reactjs/reactjs_introduction.htm.

[7] Google, Sep 2025. [Online]. Available: https://angular.dev/overview.

[8] GeeksForGeeks, Sep 2025. [Online]. Available: https://www.geeksforgeeks.org/jquery/jquery-tutorial/.

[9] jQuery Foundation, Sep 2025. [Online]. Available: https://jquery.com/.

[10] Node.js Foundation, Sep 2025. [Online]. Available: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs.

[11] Django Software Foundation, Sep 2025. [Online]. Available: https://www.djangoproject.com/.

[12] GeeksForGeeks, Sep 2025. [Online]. Available: https://www.geeksforgeeks.org/blogs/backend-development/.

[13] GeeksforGeeks C#, [Online]. Available: https://www.geeksforgeeks.org/c-sharp/introduction-to-c-sharp/.

[14] GeeksforGeeks C#, [Online]. Available: https://www.geeksforgeeks.org/c-sharp/introduction-to-net-framework/.

[15] Ruby, [Online]. Available: https://www.ruby-lang.org/en/.

[16] RubyOnRails, [Online]. Available: https://rubyonrails.org/.

[17] IBM, [Online]. Available: https://www.ibm.com/think/topics/java-spring-boot.

[18] IBM, Sep 2025. [Online]. Available: https://www.ibm.com/cloud/learn/nosql-databases.

[19] Amazon Web Services, Sep 2025. [Online]. Available: https://aws.amazon.com/nosql/.

[20] Oracle, Sep 2025. [Online]. Available: https://www.oracle.com/mysql/what-is-mysql.

[21] AppsCode, Sep 2025. [Online]. Available: https://appscode.com/blog/post/mysql-and-its-use-cases.

[22] Microsoft, Sep 2025. [Online]. Available: https://www.microsoft.com/en-us/sql-server.

[23] DevOpsSchool, Sep 2025. [Online]. Available: https://www.devopsschool.com/blog/what-is-sql-server-and-use-cases-of-sql-server.

[24] Firebase, 2024. [Online]. Available: https://firebase.google.com/.

[25] CData, 2023. [Online]. Available: https://www.cdata.com/blog/firebase-use-cases.

[26] MongoDB Inc., Sep 2025. [Online]. Available: https://www.mongodb.com/.

[27] CData Software, Sep 2025. [Online]. Available: https://www.cdata.com/blog/mongodb-use-cases.

[28] Apache Software Foundation, Sep 2025. [Online]. Available: https://cassandra.apache.org.

[29] Redis Ltd., Sep 2025. [Online]. Available: https://redis.io/learn/howtos/analytics.

[30] System Design Newsletter, Sep 2025. [Online]. Available: https://newsletter.systemdesign.one/p/redis-use-cases.

[31] MariaDB Foundation, Sep 2025. [Online]. Available: https://mariadb.org.

[32] Estuary, Sep 2025. [Online]. Available: https://estuary.dev/blog/best-open-source-databases.

[33] Google Firebase, Feb 2025. [Online]. Available: https://firebase.google.com/products/auth.

[34] Mozilla, Sep 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API.

[35] "eventbrite," [Online]. Available: https://www.eventbrite.com/organizer/overview/.

[36] Alphansotech, [Online]. Available: https://www.alphansotech.com/blog/how-eventbrite-works-guide-to-use-event-management-platform/. [Accessed 4 Oct 2026].

[37] "cevent," [Online]. Available: https://www.cvent.com.

[38] Whova, [Online]. Available: https://whova.com/pages/whova-app-exhibitor-guide/. [Accessed Sep 2025].

[39] Bizzabo, [Online]. Available: https://www.linkedin.com/products/bizzabo/. [Accessed Sep 2025].

[40] Accelevents, [Online]. Available: https://www.accelevents.com/faqs. [Accessed 26 Sep 2025].

[41] Accelevents, [Online]. Available: https://www.accelevents.com/blog/best-event-management-software. [Accessed 26 Sep 2025].

[42] RingCentral, [Online]. Available: https://www.ringcentral.com/rc-events/solutions/virtual-event-platform.html. [Accessed 26 Sep 2025].

[43] EventXPro, [Online]. Available: https://www.eventxpro.com/. [Accessed 26 Sep 2026].

[44] Odoo, [Online]. Available: https://www.odoo.com/app/events. [Accessed 26 Sep 2026].

[45] Whova, [Online]. Available: https://whova.com/faq/what-is-whova/. [Accessed Sep 2025].

[46] Eventtia, 2025. [Online]. Available: https://www.eventtia.com/en/best-swoogo-competitors-and-alternatives/.
[47] Bizzabo, [Online]. Available: https://www.bizzabo.com/. [Accessed Sep 2025].
[48] [Online]. Available: https://wproom.com.
[49] [Online]. Available: https://www.shutterstock.com/search/javascript-logo?image_type=vector&dd_referrer=https%3A%2F%2Fwww.google.com%2F.
[50] [Online]. Available: https://commons.wikimedia.org/wiki/File:React-icon.svg.
[51] [Online]. Available: https://fi.wikipedia.org/wiki/Angular.
[52] [Online]. Available: https://medium.com/@alexefimenko/jquery-4-a-new-era-3695332777ef.
[53] [Online]. Available: https://en.wikipedia.org/wiki/Node.js.
[54] [Online]. Available: https://en.wikipedia.org/wiki/.NET_Framework.
[55] [Online]. Available: https://www.fullstackpython.com/django.html.
[56] [Online]. Available: https://en.wikipedia.org/wiki/Laravel.
[57] [Online]. Available: https://en.wikipedia.org/wiki/Ruby_on_Rails.
[58] [Online]. Available: https://www.clipartmax.com/middle/m2H7Z5K9N4K9Z5Z5_spring-framework-logo-svg-png-download-java-spring/.
[59] [Online]. Available: https://www.fullstackpython.com/mysql.html.
[60] [Online]. Available: https://greenwireit.com/windows-server/microsoft-sql-server-express-windows-firewall-dynamically/.
[61] [Online]. Available: https://en.wikipedia.org/wiki/Apache_Cassandra.
[62] [Online]. Available: https://knowledgehive.co.in/courses/mariadb/.
[63] [Online]. Available: https://www.skiplevel.co/blog/part-2-rest-api-components-how-to-read-them.
[64] [Online]. Available: https://www.iconpacks.net/free-icon/rest-api-blue-logo-22098.html#google_vignette.
[65] [Online]. Available: https://evgenyzborovsky.com/2018/03/26/firebase-authentication-in-xamarin-forms/.
[66] [Online]. Available: https://apps.zid.sa/application/16.
[67] [Online]. Available: https://commons.wikimedia.org/wiki/File:Google_Maps_Logo_2020.svg.
[68] [Online]. Available: https://apidog.com/blog/websocket-testing-tools/.
[69] [Online]. Available: https://tutorialsight.com/websockets-vs-http/.
[70] [Online]. Available: https://commons.wikimedia.org/wiki/File:MongoDB_Logo.svg.
[71] [Online]. Available: https://www.logo.wine/logo/Redis.
[72] [Online]. Available: https://en.wikipedia.org/wiki/Bootstrap_%28front-end_framework%29.