

# MAX FIT EVENT MANAGEMENT APPLICATION USING SALESFORCE

Project report submitted in partial fulfilment of the requirement for the degree of  
Bachelor of Technology  
in  
**Computer Science and Engineering/Information Technology**

By  
(AMAN KUMAR (191419))

Under the supervision  
of

(Dr. Sunil Datt Sharma)

**Associate Professor**

(Dr. Yugal Kumar)

**Associate Professor**

to



Department of Computer Science & Engineering and Information Technology  
**Jaypee University of Information Technology Wahnaghat, Solan-173234**  
**Himachal Pradesh**

## Declaration

I hereby declare that the work presented in this report entitled “**Max Fit Event Management Application using Salesforce**” in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2023 to May 2023 under the supervision of (Dr. Sunil Datt Sharma) (Assistant Professor,SG (ECE))and Dr.Yugal Kumar(Associate Professor,(CSE)).I also authenticate that I have carried out the above mentioned project work under the proficiency stream **Cloud Computing**.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)  
Aman Kumar  
191419

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor 1 Signature)  
Dr. Sunil Datt Sharma  
Associate Professor  
ECE  
Dated:

(Supervisor 2 Signature)  
Dr. Yugal Kumar  
Associate Professor  
CSE  
Dated:

**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT**  
**PLAGIARISM VERIFICATION REPORT**

Date: .....

Type of Document (Tick):  PhD Thesis  M.Tech Dissertation/ Report  B.Tech Project Report  Paper

Name: \_\_\_\_\_ Department: \_\_\_\_\_ Enrolment No \_\_\_\_\_

Contact No. \_\_\_\_\_ E-mail. \_\_\_\_\_

Name of the Supervisor: \_\_\_\_\_

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): \_\_\_\_\_

**UNDERTAKING**

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

**Complete Thesis/Report Pages Detail:**

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

**FOR DEPARTMENT USE**

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

**FOR LRC USE**

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"> <li>• All Preliminary Pages</li> <li>• Bibliography/Images/Quotes</li> <li>• 14 Words String</li> </ul>		Word Counts	
<b>Report Generated on</b>			Character Counts	
		<b>Submission ID</b>	Total Pages Scanned	
			File Size	

Checked by  
Name & Signature

Librarian

**Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at [plagcheck.juit@gmail.com](mailto:plagcheck.juit@gmail.com)**

## ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor **Dr. Sunil Datt Sharma**, Associate Professor Department of ECE and **Dr. Yugal Kumar**, Associate Professor (CSE), Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of “**Cloud Computing**” to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude **Mr. Ghanshyam Bhatt (Director of Technology)**, **Ms. Megha Goel (Project Manager)** and **Mr. Deepak Kumar Singh (Project Mentor)** of Mirketa Inc for their kind helps to finish my project.

I would also generously welcome each one of those individuals who have helped us straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non- instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Aman Kumar  
(191419)

## Table of Content

<b>Title</b>	<b>Page No.</b>
<b>Certificate</b>	<b>I</b>
<b>Plagiarism Certificate</b>	<b>II</b>
<b>Acknowledgement</b>	<b>III</b>
<b>Table of Content</b>	<b>IV</b>
<b>List of Figures</b>	<b>V-VI</b>
<b>Abstract</b>	<b>VII</b>
<b>Chapter-1 (Introduction)</b>	<b>1-12</b>
<b>Chapter-2 (Literature Survey)</b>	<b>13-22</b>
<b>Chapter-3 (System Design, Analysis/Design/Development/Algorithm)</b>	<b>23-44</b>
<b>Chapter-4 (Performance Analysis)</b>	<b>45-52</b>
<b>Chapter-5 (Conclusion)</b>	<b>53-58</b>
<b>References</b>	<b>59</b>

## List of Figures

Figure no.	Title
3.1	Salesforce-entity relationship
3.2	Location
3.3	Events
3.4	Attendees
3.5	Speaker
3.6	Error log
3.7	Event-Speaker
3.8	Event-Attendee
3.9	Event Organizer
3.10	Validation Rules
3.11	Validation Rule on Event Attendee
3.12	Validation Rule on Event-Speaker
3.13	Attendee Rule
3.14	Event Organizer Rule
3.15	Speaker Rule
3.16	Object Permission setup
3.17	Database Batchable Class
3.18	Error Log Record Class
3.19	Event Attendee Trigger
3.20	Event Attendee Trigger
3.21	Event Speaker Trigger
3.22	Event Registration Controller Class for Visualforce Page
3.23	Event Attendee Controller Class for Visualforce Page
3.24	Event Registration Form
3.25	Attendee Registration Form

<b>3.26</b>	Speaker Registration Controller Class for Visualforce Page
<b>4.1</b>	Recurring_validation_rule
<b>4.2</b>	Virtual_Rule
<b>4.3</b>	date time validation rule
<b>4.4</b>	Event_Type_Rule
<b>4.5</b>	Attendee rule
<b>4.6</b>	Speaker rule
<b>4.7</b>	event organizer rule
<b>4.8</b>	Event Speaker Trigger
<b>4.9</b>	Event Attendee Trigger
<b>4.10</b>	Event Registration Controller Page
<b>4.11</b>	Speaker Registration Controller Page
<b>4.12</b>	Attendee Registration Controller Page

## **Abstract**

MAX FIT, an event management company, is seeking an efficient software solution to enhance their event management capabilities, particularly in managing attendee information and event locations. As a developer, the task has been assigned to develop a solution utilizing the Salesforce Platform. This abstract provides a comprehensive overview of the project, outlining the objectives, requirements, and potential benefits of the proposed software solution. The primary goal of the project is to create an event management software system that maximizes the efficiency and effectiveness of MAX FIT's event planning and execution processes. The solution will be built on the Salesforce Platform, leveraging its robust features and functionalities for seamless integration with MAX FIT's existing Salesforce ecosystem. Salesforce offers a scalable and customizable platform, providing a solid foundation for the development of a tailored event management system. The proposed software solution will address key requirements identified by MAX FIT. It will empower the company to efficiently manage event-related data, including attendee information and event locations. The system will enable the seamless registration and tracking of attendees, facilitating streamlined communication and engagement throughout the event lifecycle. Additionally, it will provide comprehensive tools for managing event locations, including venue selection, logistics planning, and resource allocation.

In conclusion, the development of an event management software solution using the Salesforce Platform will significantly enhance MAX FIT's ability to manage events effectively. By leveraging Salesforce's robust features, the solution will streamline attendee management and event location logistics, resulting in improved efficiency, accuracy, and collaboration within the organization. With the proposed system in place, MAX FIT will be better positioned to deliver exceptional event experiences and drive their business forward.



# **Chapter-1**

## **1.1 Introduction**

In today's cutthroat corporate environment, successful sales management is critical. Leading CRM platform Salesforce Sales Cloud revolutionizes sales processes. Sales Cloud gives sales teams the tools they need to increase productivity, solidify client connections, and enhance overall sales success. Sales Cloud serves as a hub, allowing salespeople to effectively manage the whole sales cycle. It offers solutions for streamlining sales tasks, such as lead monitoring, opportunity management, revenue forecasting, teamwork, and performance analysis. The lead management features of Sales Cloud are noteworthy. Businesses can get leads from a variety of sources and compile them in one central database. So that the transition from marketing to sales is as smooth as possible, sales people can then rank and assign prospects. As a result of the platform's comprehensive picture of leads, teams may concentrate on high-potential opportunities and increase conversion rates. Sales Cloud's forte is opportunity management. Users can collaborate with team members, customize sales procedures, and visually track and manage sales possibilities. This accelerates contract closure by streamlining processes and enhancing communication.

Strong reporting and analytics features in Salesforce offer insightful data on important sales indicators. Sales managers can make data-driven choices, pinpoint areas for development, and anticipate income precisely thanks to real-time dashboards and reports. Salesforce products and outside programmes are effortlessly integrated with Sales Cloud. A single sales ecosystem adapted to particular business demands is produced by this integration. Sales Cloud serves as a central center, providing a comprehensive approach to sales administration.

## **1.2 Problem Statement**

For the event management system to guarantee data accuracy, consistency, and appropriate access control, the present setup for Validation Rules, Duplicate Rules, Profiles, Users, Roles, and Sharing Rules has to be improved. The functioning and user experience are being impacted by a number of problems that have been found. The following issues need to be resolved:

### **Milestone 1:**

#### **Validation Rule Setup:**

1.1. The requirement that the Frequency field be filled in if the "Recurring?" checkbox is ticked should be enforced by the validation rule for the Event object. On the other hand, the checkbox should not be checked if you don't want the Frequency field to be selectable.

1.2. Users shouldn't be able to choose a location on the Event record if the Event Type field is set to "Virtual" because of the validation rule.

1.3. If a value is entered in the End Date/Time field, the validation rule should make sure that it is at least one day before the Start Date/Time.

1.4. The validation rule should require users to choose a Location on the Event record when the Event Type field is set to "In-Person."

#### **Event Attendee Object:**

2.1. Only Events with a future End Date, an event live checkbox, and an accepted event (Remaining Seats field value not 0) should have attendees linked with them.

#### **Event Speaker Object:**

3.1. Only Events with a future End Date and the Event Live checkbox selected should have Speakers attached to them.

### **Milestone 1.2 - Duplicate Rule Setup:**

- For the Speaker object, duplicate records with the same Email and Phone should not be permitted.
- For the Attendee object with the same Name, Email, and Phone, duplicate records shouldn't be permitted.
- To avoid duplicate Organisers in the system, use the same duplicate rule that is used for Speaker objects.

### **Milestone 1.3 - Profile, User, OWD, and Role Setup:**

1.3 Profile Setup: In order to construct the three required profiles (Event Organiser, Event Attendee, and Speaker), the Standard User profile must be copied.

#### **User Setup:**

1.3.2.1. For testing purposes, users must be created.

#### 1.3.3. Role Setup:

1.3.3.1. There should be created positions (Organiser, Attendee, and Speaker), and each function should report to the CEO.

#### 1.3.3.2. **Role Hierarchy needs to be set up as follows:**

CEO

- Organizer
- Attendee
- Speaker

#### **Sharing Rule Setup:**

Implementing sharing rules will ensure that the Organiser role has access to every Speaker and Attendee record. The following information is needed for two sharing rules:

Speaker Object: Make a sharing rule that grants read/edit access to Speaker records for the Organiser role.

Attendee Object: To share Attendee records with the Organiser role and give read/edit access, create a sharing rule.

By fixing these issues, the event management system will perform more effectively and have more reliable data as well as better access control and user experience.

## **Milestone 2**

Apex Class Development: Create a reusable Apex class that has a function to insert the entries from the error log. The parameters for this method must be included in order to obtain the dynamic information for the fields (Log Date/Tile, Log Details, and Process Name).

## **Milestone 3**

Event-Speaker Object Trigger Development - Create a trigger on the event-speaker object that will generate an error if the speaker selected on the event-speaker record already has an event associated with him. For a speaker, only one event will be taking place at once. Abandon Duplicate Bookings.

- Which object will serve as the Trigger (Event - Speaker)
- What are the pre-update and pre-insert events?

Output - Check for duplicate bookings in the output and throw an error

## **Milestone 4**

Trigger Development - Develop an Apex trigger on the Event Attendee object (Whenever a New Record is Created) to send the Attendee an email confirming their registration. Use the email format shown below.

Subject: Pass for the "Event name here"

Message Body: Dear Name of Attendee, We appreciate your interest in "Event Name Here," which will take place at "Event Location Here" on "Event Date Here." See you at the event, we can't wait to have you.

Gratitude to "Organiser Name Here"

### **Milestone 5**

Develop Apex Batch - Create an Apex batch - Create an Apex batch that should erase all organised event records that are more than two months old. Use the Event Record's Live? and End Date of the Event checkboxes as a hint.

Send an email to admin (your email address) stating that the batch execution has been completed in the batch apex's finish method.

### **Milestone 6**

Develop Unit Test - Build a unit test that covers at least 90% of the code in the aforementioned class, trigger, and batch.

Must Have -

- Minimum 90% code coverage
- Use of Test.startTest & Test.stopTest, @TestSetup annotation, and
- Asserts Methods
- A test case for a negative test case must exist.
- Each test case must be passed without errors.
- Optional use of the TestUtility Class

## **Milestone 7**

Create a registration form for the following collapsed sections:

**Event organiser and related fields; Event object and related fields; Event location object and related fields**

A record for the event organiser, the event, and the venue should be produced after saving.

## **Milestone 8**

Make the following fields mandatory on any form you create to register speakers:

- Speaker object, Speaker, and lookup field to choose an event
- A record for the speaker and event-speaker should be created after saving.

## **Milestone 9**

- Make an attendance registration form with the following parts that can be collapsed:
- Lookup field to choose an event; Attendee Details; Attendee object fields
- Location object fields for the attendee location
- An event-attendee record should be created after saving the data.

## 1.3 Objectives

The objective of this project is to enhance an event management system by addressing various problem areas and implementing new functionalities. The project is divided into multiple milestones, each with its own objectives. The overall project objective is to improve the functionality, data integrity, user experience, and efficiency of the event management system. The specific objectives for each milestone are as follows:

### **Milestone 1: Validation Rule Setup**

Implement validation rules for the Event object to enforce data integrity and user input requirements. Implement validation rules for the Event Attendee and Event Speaker objects to ensure proper associations with future events.

### **Milestone 1.2: Duplicate Rule Setup**

Establish duplicate rules to prevent the creation of duplicate records for Speakers, Attendees, and Organizers.

### **Milestone 1.3: Profile, User, OWD, and Role Setup**

Clone profiles from the Standard User profile and create profiles for Event Organizer, Event Attendee, and Speaker.

Set up users for testing purposes.

Create roles (Organizer, Attendee, and Speaker) with a hierarchical structure where all roles report to the CEO.

### **Sharing Rule Setup**

Implement sharing rules to ensure Speaker and Attendee records are shared with the Organizer role, granting appropriate read/edit permissions.

### **Milestone 2: Apex Class Development**

Develop a reusable Apex Class with a method to insert Error Log Object records dynamically.

### **Milestone 3: Trigger Development (Event - Speaker Object)**

Develop a trigger on the Event - Speaker object to prevent duplicate bookings and enforce a one-event-per-speaker restriction.

#### **Milestone 4: Trigger Development (Event Attendee Object)**

Develop an Apex Trigger on the Event Attendee object to send confirmation emails to attendees upon successful registration.

#### **Milestone 5: Apex Batch Development**

Write an Apex Batch class to delete event records that are more than 2 months old and have been organized. Send an email notification to the admin upon the completion of batch execution.

#### **Milestone 6: Unit Test Development**

Develop comprehensive unit tests for the class, trigger, and batch classes with at least 90% code coverage. Use appropriate annotations, utility classes, and assert methods to ensure successful test execution.

#### **Milestone 7: Event Registration Form Development**

Create a form for event registration with collapsible sections, capturing event organizer, event details, and event location information.

Upon form submission, create records for the event organizer, event, and location in the system.

#### **Milestone 8: Speaker Registration Form Development**

Create a form for speaker registration, capturing speaker details and allowing selection of an associated event. Upon form submission, create records for the speaker and event-speaker association.

#### **Milestone 9: Attendee Registration Form Development**

Create a form for attendee registration with collapsible sections, capturing attendee details and attendee location information. Allow selection of an associated event. Upon form submission, create records for the attendee and event-attendee association. The project aims to achieve these objectives while maintaining a high level of code coverage, utilizing best practices in testing, and providing a user-friendly registration experience for event organizers, speakers, and attendees.



## 1.4 Methodology

The project will employ an Agile methodology, specifically Scrum, to ensure iterative development, collaboration, and adaptability. The methodology will be tailored to accommodate the unique milestones and tasks involved in the project. The following steps outline the project methodology:

### **Project Initiation:**

Gather project requirements and objectives.

Identify stakeholders and establish communication channels.

Define project milestones and prioritize tasks.

### **Sprint Planning:**

Break down the project into smaller tasks for each milestone.

Estimate the effort and duration for each task.

Establish acceptance criteria for each task.

Prioritize tasks based on dependencies and criticality.

Assign tasks to team members.

### **Sprint Execution:**

Conduct daily stand-up meetings to discuss progress, challenges, and plans.

Collaboratively develop validation rules, duplicate rules, profiles, triggers, and other components.

Implement sharing rules and role hierarchy according to requirements.

Develop a reusable Apex class for error logging.

Create a trigger for the Event-Speaker object to prevent duplicate bookings.

Develop a trigger for the Event Attendee object to send confirmation emails.

Write an Apex Batch class to delete old event records.

Develop comprehensive unit tests with a minimum of 90% code coverage.

Design and create registration forms for the event organizer, speaker, and attendee.

**Sprint Review and Retrospective:**

Review completed tasks against acceptance criteria.

Demonstrate functionality to stakeholders and gather feedback.

Conduct a retrospective to identify areas for improvement in the next sprint.

Update the project plan based on feedback and changes.

Iterative Development:

Repeat steps 2-4 for subsequent milestones.

Continuously refine and adjust the project plan based on progress and feedback.

**Project Closure:**

Perform final testing and quality assurance checks.

Deploy the developed components to the production environment.

Conduct a final review to ensure all objectives have been met.

Prepare project documentation, including user guides and release notes.

Notify stakeholders about project completion and provide necessary training and support.

Throughout the project, regular communication and collaboration among team members, stakeholders, and users will be maintained. Any changes or issues that arise during development will be addressed through ongoing discussions and adjustments to the project plan. The ultimate aim is to deliver a high-quality event management system that fulfills the specified requirements and delivers a seamless user experience.

## 1.5 Organization

To ensure efficient task management and execution, the project can be structured as follows:

### **Project Manager:**

Responsible for overseeing the project, including planning, coordination, and execution. Manages the project timeline, milestones, and deliverables.

Facilitates communication between stakeholders and the development team.

Ensures the project objectives are achieved and meets stakeholders' expectations.

### **Development Team:**

Apex Developers: Responsible for developing Apex classes, triggers, and batch processes based on the requirements.

Salesforce Administrator: Handles profile setup, role configuration, sharing rules, and other Salesforce-related tasks.

### **Quality Assurance/Testers:**

Conduct unit testing, ensure code coverage, and perform quality checks to ensure functionality and stability.

Stakeholders:

### **CEO:**

Provides high-level guidance and project approval.

### **Event Organizers:**

Collaborate on setting up validation rules, profiles, and sharing rules to meet specific requirements.

### **Event Attendees:**

Expect a seamless registration process and confirmation emails upon successful registration.

**Speakers:**

Require a smooth registration process and prevention of duplicate bookings.

**Regular meetings:**

Conduct weekly or bi-weekly meetings with the development team to review progress, address challenges, and plan upcoming tasks. Stakeholder meetings: Arrange periodic meetings with stakeholders to provide updates, gather feedback, and discuss any changes or new requirements.

**Collaboration tools:**

Utilize project management tools like Jira for task tracking, document sharing platforms such as Google Drive or SharePoint for documentation, and communication tools like Slack or Microsoft Teams for real-time collaboration and communication.

**Project Documentation:**

Create a detailed project plan that outlines milestones, tasks, timelines, and responsible individuals. Document functional and technical specifications for each milestone/task. Maintain a central repository for project-related documents, including design documents, test cases, and user guides. Regularly update documentation as changes occur and ensure it is easily accessible to the project team. By establishing a clear project organization, promoting effective communication, and maintaining comprehensive documentation, the project can progress smoothly. Collaboration and feedback between the team and stakeholders will ensure prompt resolution of challenges and successful delivery of a high-quality event management system that fulfills the specified requirements.

## **Chapter 2**

### **Literature Survey**

Salesforce which is a type of customer relationship management (CRM) platform built in the cloud that provides a variety of tools and applications for companies in a variety of industries. Salesforce has significantly grown in popularity in recent years and has established itself as the go-to option for event management because of its extensive features and versatility. With a focus on its benefits, drawbacks, and general efficacy, this study of the literature examines the body of knowledge about Salesforce's use for event management.

#### **The advantages of Salesforce for Event Management:**

##### **Centralized Data Management:**

One of Salesforce's main benefits for event management is centralised data management. Event planners may effectively manage many parts of event planning and execution while preserving accurate and consistent data throughout the event lifecycle by making use of Salesforce's CRM features.

The ability to efficiently manage attendance data is one of the main advantages of Salesforce's centralised platform. A central repository can be used by event planners to store and organise attendee data, such as contact information, preferences, and registration history. The information on participants may be easily accessed through this central database, giving event planners a complete picture of their attendees.

Salesforce supports another essential component of event management: registration tracking. With Salesforce, event planners can design personalised registration forms and collect participant registrations within the system itself. This speeds up the registration process and makes sure that all participant data is gathered uniformly.

Additionally, Salesforce provides automation tools like email alerts and confirmation messages to improve the registration process for visitors.

### **Enhanced Attendee Engagement:**

Using Salesforce to handle events has many advantages, including increased participant engagement. Event planners may interact with participants in a personalised and effective way using a variety of engagement tools from Salesforce, which will eventually increase attendee satisfaction and encourage better levels of participation during the event.

Salesforce offers personalised emails as one of its main interaction strategies. Event planners can send personalised and targeted emails to participants using the platform's built-in email functionality. These emails can provide each guest with a customised experience by including personalised greetings, event updates, session suggestions, and unique offers. Event planners can promote a feeling of exclusivity and make guests feel valued by customising the messaging and content to their preferences and interests. This will enhance engagement and help attendees feel more connected to the event.

Salesforce offers another effective element for boosting attendee engagement: social media integration. To generate awareness about the event and promote attendance, event planners can make use of social media sites like Twitter, Facebook, and LinkedIn. Salesforce allows seamless connectivity with major social media networks, enabling event planners to plan and publish posts, advertise event-related content, and communicate with attendees in real-time. The reach and impact of the event can be significantly increased by actively involving attendees on social media to create enthusiasm, allow networking possibilities, and encourage sharing of experiences.

### **Real-time Reporting and Analytics:**

Salesforce's real-time reporting and analytics capabilities are crucial components that let event planners obtain insightful information, gauge the effectiveness of their events, and make decisions based on data-driven

analysis. Salesforce provides event planners with comprehensive views of key performance indicators (KPIs) and pertinent metrics through its powerful reporting capabilities and real-time dashboards.

Tracking registration numbers is one of the main benefits of Salesforce's reporting features. The amount of registrations received, registration trends over time, and registration data comparisons across other events or event sessions can all be seen in reports that event organisers can produce. With the use of these data, event planners can better understand the levels of interest and engagement among attendees, which will help them allocate resources and manage the event's marketing effectively.

#### **Automation and Workflow Efficiency:**

One of the main benefits of using Salesforce for event management is automation and workflow effectiveness. Event planners may streamline repetitive operations, reduce manual labour, and improve overall event management procedures with the platform's comprehensive automation tools.

The handling of registrations is one of the main areas where automation is advantageous. Salesforce has the potential to completely automate the registration process. The automated workflows that gather participant data, provide customised registration confirmations, and distribute event tickets can be set up by event planners. Organisers greatly reduce the possibility of mistakes and delays by automating these activities, which also eliminates the need for manual data entry. By offering a simple and straightforward registration process, this not only increases efficiency but also improves the attendance experience.

Additionally, Salesforce's automation capabilities include procedures for communication. Event planners can use automated email workflows to deliver communication messages like post-event surveys, reminders about the event,

and registration confirmations. These workflows can be activated in response to particular events or goals, ensuring that attendees are given timely and pertinent information. By giving pertinent updates and reminders, event organisers may boost attendee engagement while saving time and maintaining a consistent message.

### **Integration with Third-party Tools:**

Enhancing the event management process is made possible by Salesforce's ability to integrate with outside tools. Salesforce enables easy connectivity with a variety of technologies, including email marketing programmes, survey tools, event ticketing programmes, and more. This integration gives event planners the ability to use more functionalities and increase the capabilities of their event management system, leading to an event management process that is more thorough and effective.

Making it possible to generate personalised and targeted email campaigns is one of the main advantages of connecting Salesforce with email marketing solutions. Event planners may easily connect their Salesforce data to well-known email marketing tools like Mailchimp or Constant Contact. With the help of this integration, event planners can divide their attendance list into groups according to preferences for particular sessions, registration types, or demographics. Using the integration, organisers may target certain groups of attendees with emails that are both personalised and pertinent. This focused approach improves the efficiency of email marketing initiatives, leading to greater engagement and increased attendance rates. Event planners have a great tool for gathering feedback and assessing guest satisfaction when survey platforms are integrated with Salesforce.

Platforms like SurveyMonkey or Typeform can be integrated with Salesforce by event planners to automatically collect survey responses and sync the information with attendance records. This integration makes it easier to collect participant feedback, giving event planners useful information about attendees' preferences for sessions and overall event satisfaction. Because of the



integrated data in Salesforce, event planners may examine survey results along with other attendance data, enabling data-driven decision-making and ongoing event development.

Integration with event ticketing systems is another essential feature of Salesforce's third-party integration capabilities. Salesforce can be easily linked with well-known ticketing services like Eventbrite or Ticketmaster, giving event planners the ability to control ticket sales, monitor attendance, and create data all from one place. Organisers can see the status of attendee registration, ticket sales, and event capacity in real time by integrating ticketing systems with Salesforce. By streamlining the ticketing process and enabling automated updates to attendee records, this interface gives event planners access to detailed information on event attendance and revenue.

### **Challenges and Considerations:**

#### **Complexity of Configuration:**

Implementing Salesforce for event management might create complications that call for specialised knowledge and skills. Although Salesforce provides a powerful platform, tailoring it to fit particular event requirements frequently necessitates the intervention of knowledgeable Salesforce administrators or developers.

It might be difficult to set up Salesforce for event management without a thorough understanding of the platform's architecture and customization possibilities. Event planners may have certain needs that go above and beyond Salesforce's default functionality. Customization entails shaping the platform to adhere to the distinct business rules, data structures, and processes of the event management process. Salesforce setup skills is needed for this, including the ability to create unique objects, fields, workflows, and validation rules. Additionally, Apex, the programming language used by Salesforce, may be required to provide bespoke functionality or connections with external systems.

Additionally, due to the variety of events and the requirements that go along with them, configuration is complicated. Size, format, target audience, and objectives can all vary for events. The requirements for session management, registration workflows, data tracking, and reporting can vary depending on the event. An in-depth knowledge of both event management procedures and Salesforce's capabilities is necessary to configure Salesforce to take into account these special elements. It entails developing new objects and fields, translating event-specific requirements to Salesforce's data model, and implementing the required automation rules.

### **Cost Considerations:**

Cost-wise, licence fees make up a considerable portion. To accommodate varied business demands, Salesforce provides a range of versions and price options. The right edition must be chosen by organisations based on an evaluation of their unique requirements and demands for event management. User count, desired functionality, and required level of support are just a few examples of the variables that affect licencing costs. To achieve cost effectiveness, it is essential to thoroughly assess the features offered in each edition and determine whether they satisfy the organization's event management needs.

It's important to take customization costs into account. While Salesforce has several tools for event management that are available right out of the box, some organisations may have particular needs that call for customization. Develop custom objects, fields, workflows, or integrate with external systems as part of the customization process. Depending on how intricate the customization is, businesses may need to set aside resources, such as Salesforce administrators or developers, to set up and modify the platform to suit their particular requirements. The total budget needs to account for the potential additional expenses associated with these customisation initiatives, such as additional development time or the employment of outside experts.

In conclusion, organisations should consider the costs of licencing, customization, and continuing maintenance when deciding whether to use Salesforce for event management.

To make sure the platform adheres to particular event management requirements, it is essential to thoroughly evaluate the features and capabilities offered by various licencing editions and take customisation requests into account. Organisations should also budget resources appropriately for ongoing maintenance expenses. The value and financial sustainability of deploying Salesforce for event management, however, should be assessed in connection to cost concerns and predicted advantages and ROI.

### **User Adoption and Training:**

For Salesforce to be successfully implemented for event management, user adoption and training are crucial. Organisations must spend heavily on thorough training programmes when switching to a new system like Salesforce to guarantee that staff employees are competent in using the platform well.

In order for Salesforce to be used for event management, users must accept and use it. This is referred to as user adoption. The adoption of the new technology and its integration into regular operations by employees are crucial. The effective adoption and use of Salesforce can be hampered by change aversion or a lack of familiarity with the platform. Organisations should put their attention on user adoption techniques that involve good communication, training, and continuing assistance to overcome these difficulties.

To give staff employees the abilities and expertise to use Salesforce for event management, thorough training programmes are essential. Training needs to be customised for each person's unique tasks and responsibilities inside the organisation. Both the fundamental Salesforce functionality and the workflows relevant to events should be covered. Combinations of techniques, such as classroom instruction, online learning, workshops, and on-the-job training, can

be used to conduct training sessions. To ensure practical comprehension and application of the learnt principles, it is crucial to provide hands-on experience and real-life scenarios.

### **Data Security and Privacy:**

When using Salesforce for event management, data security and privacy are of the utmost importance. Event planners must put in place strong security measures, abide by data protection laws, and train personnel on data handling best practises because Salesforce handles sensitive attendee data.

Event planners should use Salesforce to create the necessary security controls to guarantee data protection. To stop unauthorised access to sensitive data, this includes implementing robust access controls and authentication procedures. Only individuals who need access to the data for their duties should be granted it by using user permissions that are based on job titles and responsibilities. In order to secure data from potential breaches or unauthorised interception, encryption techniques should be used both when the data is at rest and when it is in transit.

The organisational culture and processes should incorporate data security and privacy. Event planners should develop precise policies and processes for handling data, including rules for data access, sharing, and storage. To make sure that these regulations are followed, routine audits and compliance checks should be carried out.

Event planners may protect sensitive attendee data and keep the confidence of their participants by installing strong security measures, adhering to data protection laws, and training employees on data handling best practises. Prioritising data security and privacy in Salesforce event management helps guard against potential data breaches, identity theft, and reputational harm, eventually delivering a safe and secure event experience for guests.

### **Effectiveness and Future Directions:**

While there has been some interest in the efficiency of Salesforce for event management, there hasn't been much research done on the subject. However, based on anecdotal evidence from companies that have used Salesforce, good effects are possible, including increased productivity, better attendee engagement, and better data-driven decision-making.

By reducing procedures and automating repetitive chores, Salesforce implementation for event management has demonstrated to increase efficiency. Because the platform is centralised, event planners can see a complete overview of all event-related data, which improves coordination and streamlines procedures. By minimising human labour and saving time, automation features including registration management, payment processing, and communication processes boost efficiency and productivity.

The engagement of attendees has been increased by using Salesforce's engagement solutions. Event planners are able to communicate with participants in a more personalised and targeted way thanks to personalised emails, social media integration, and mobile applications. As a result, attendees are more satisfied and participate more actively in the event, making it more engaging and dynamic. Event planners can strengthen their connections with participants, encourage networking, and simplify post-event follow-ups by utilising these engagement tools.

### **Conclusion:**

In conclusion, Salesforce offers a thorough and adaptable solution for event management that comes with a host of advantages and benefits. The platform's centralised data management features let event planners efficiently handle a range of event planning and execution tasks, such as handling attendee data, registration tracking, logistical management, and participant communication.

Through personalised emails, social media integration, and mobile applications, Salesforce can increase attendee engagement, which helps to create a unique and dynamic event experience. Increased attendance satisfaction and higher levels of participation follow from this. The ability to track important performance indicators, gauge the success of events, and make data-driven decisions is made possible by real-time reporting and analytics technologies. Real-time dashboards are a significant resource for gathering information on registration trends, attendance demographics, and session popularity. This information enables event planners to make quick modifications and improve their event plans.

By automating time-consuming procedures like registration, payment processing, and communication workflows, Salesforce's automation capabilities streamline the process of managing events. With less manual work required thanks to automation, event planners are free to concentrate on more strategic and worthwhile tasks.

Another significant benefit of Salesforce is its ability to integrate with third-party applications, giving event planners access to more features and enhancing the functionality of their event management platform. Event planners can increase their event offerings and boost attendees' experiences at events by seamlessly integrating with email marketing tools, survey platforms, event ticketing systems, and other third-party services.

When using Salesforce for event management, organisations should be aware of some issues. Due to the complexity of platform configuration, specialised knowledge and the participation of skilled Salesforce administrators or developers may be necessary. It is important to carefully weigh the associated costs against the anticipated returns on investment, including licencing fees, customization charges, and continuing maintenance.

## Chapter-3

### System Development

#### **Design:**

The project involves the development and enhancement of an event management system through various milestones. These milestones aim to improve functionality, data integrity, and user experience. The key objectives and tasks for each milestone are summarized as follows:

Milestone 1 focuses on setting up validation rules for the Event object, Attendee object, and Speaker object. These rules ensure that events are properly configured, attendees are associated with valid events, and speakers are associated with upcoming events.

Milestone 1.2 involves establishing duplicate rules to prevent duplicate records for the Speaker object, Attendee object, and Organizers. These rules ensure that no duplicate entries are made based on specific fields such as Email, Phone, Name, and prevent multiple organizers for the same event.

Milestone 1.3 entails setting up profiles, users, and roles. Three profiles (Event Organizer, Event Attendee, and Speaker) are created, and users are generated for testing purposes. Roles are created with a hierarchical structure, and sharing rules are implemented to ensure proper access control for Speaker and Attendee records.

Milestone 2 involves the development of a reusable Apex Class that contains a method for inserting Error log Object records. This class includes dynamic parameters to capture details such as Log Date/Time, Log Details, and Process Name.

Milestone 3 focuses on developing a Trigger for the Event - Speaker object. This trigger prevents duplicate bookings by checking if a speaker is already

associated with another event. It throws an error if duplicate bookings are detected.

Milestone 4 involves creating an Apex Trigger on the Event Attendee Object. This trigger sends a confirmation email to attendees upon registration. The email includes event-specific details such as the event name, date, location, and organizer's name.

Milestone 5 entails the development of an Apex Batch to delete event records that are older than two months and have been organized. The batch is executed, and an email notification is sent to the admin upon completion.

Milestone 6 focuses on developing unit tests for the classes, triggers, and batch. The tests should achieve at least 90% code coverage and follow best practices such as using annotations, start/stop test methods, and assert statements.

Milestone 7 involves creating a form for event registration with collapsible sections for the event organizer, event details, and location information. Saving the form creates records for the event organizer, event, and location.

Milestone 8 focuses on creating a form for speaker registration, including details such as the speaker's information and a lookup field to select an event. Saving the form creates records for the speaker and the event-speaker association.

The creation of a registration form for attendees that include sections for attendance information and venue details that may be collapsed is required for Milestone 9. The form has a lookup field where you may choose an event, and when you save it, records for the attendee and the association between the attendee and the event are created.



The event management system will be improved in terms of functionality, data integrity, access control, error handling, and user experience by achieving these objectives.

## **Model Development:**

### **Object Manager:**

The Salesforce platform's standard and custom objects can be managed and customised using the Object Manager tool by administrators and developers. For operations like creating objects, changing fields, developing page layouts, specifying record types, establishing relationships between objects, and controlling object permissions, it offers a user-friendly interface.

With Object Manager, administrators and developers can create custom objects to store organization-specific data, define field properties and relationships, customise the layout of object records, establish various record types to capture various types of information and business processes, enforce data quality with validation rules, establish relationships between objects for data analysis, and manage object-level permissions to control access.

Overall, Object Manager is a strong solution that centralises object administration and customisation in Salesforce, allowing customers to customise the platform to their organization's needs.

Salesforce Entity-Relationship Diagram

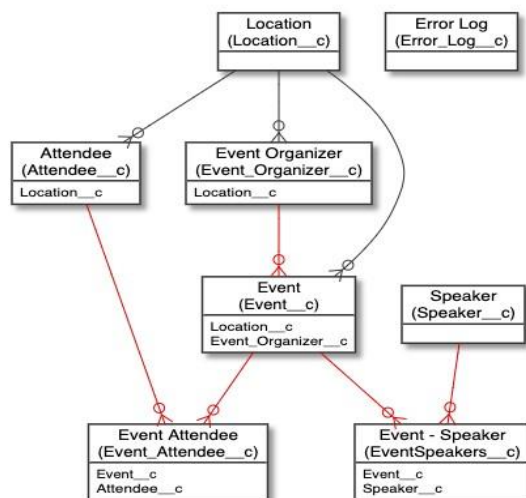


Fig 3.1: Salesforce entity-relationship

**Objects:**

In Salesforce, an object is a core component that represents a specific type of data within the platform. Objects can be categorized as either standard objects, which are provided by Salesforce, or custom objects, which are created by users to meet their specific business requirements.

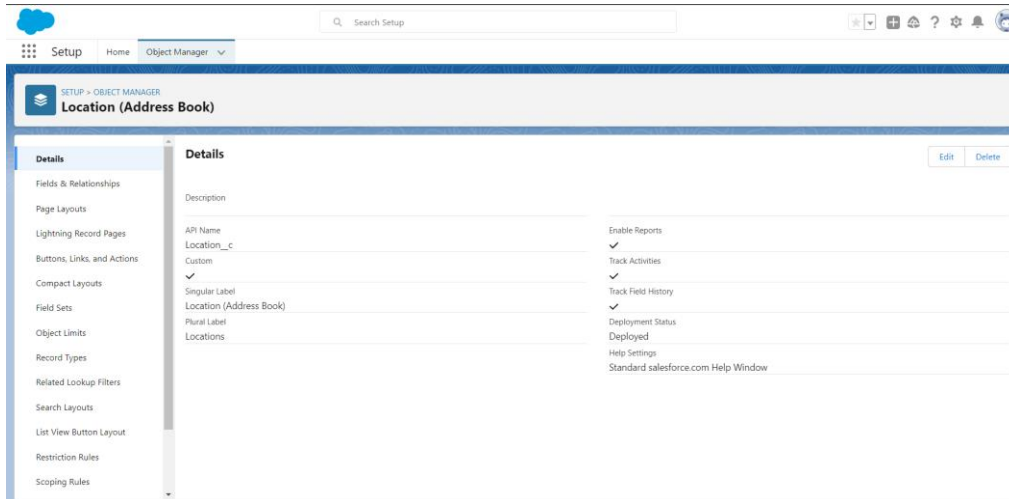
**Standard Objects:** Salesforce offers a set of predefined standard objects that are commonly used across organizations. Examples of standard objects include Accounts, Contacts, Opportunities, Leads, Cases, and Campaigns.

These objects come with predefined fields, relationships, and functionality designed to support common business processes.

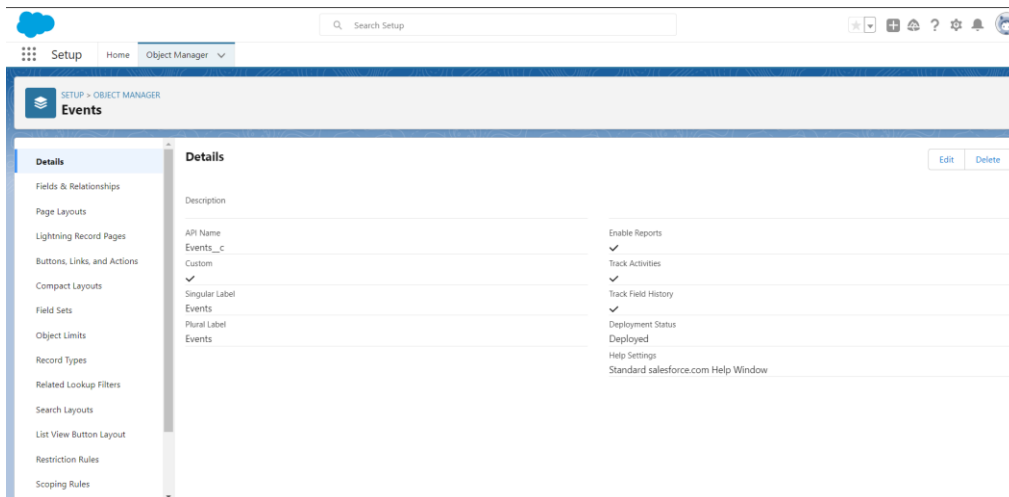
**Custom Objects:** Users have the flexibility to create custom objects tailored to their organization's unique needs. Custom objects allow users to define and store data that is specific to their business processes and not covered by standard objects. Custom fields, relationships, and business rules can be defined within custom objects.

In Max Fit application, we have used following objects to achieve our project objectives:

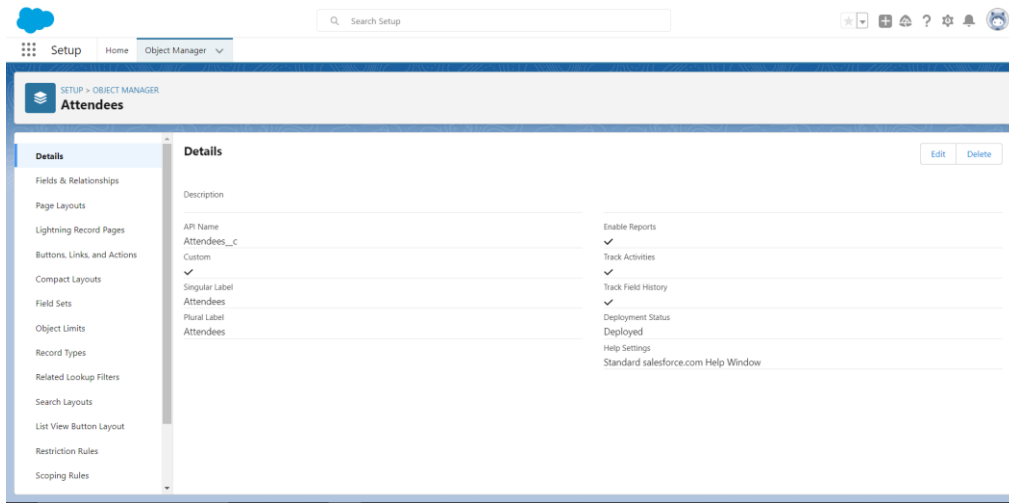
### Custom Objects:



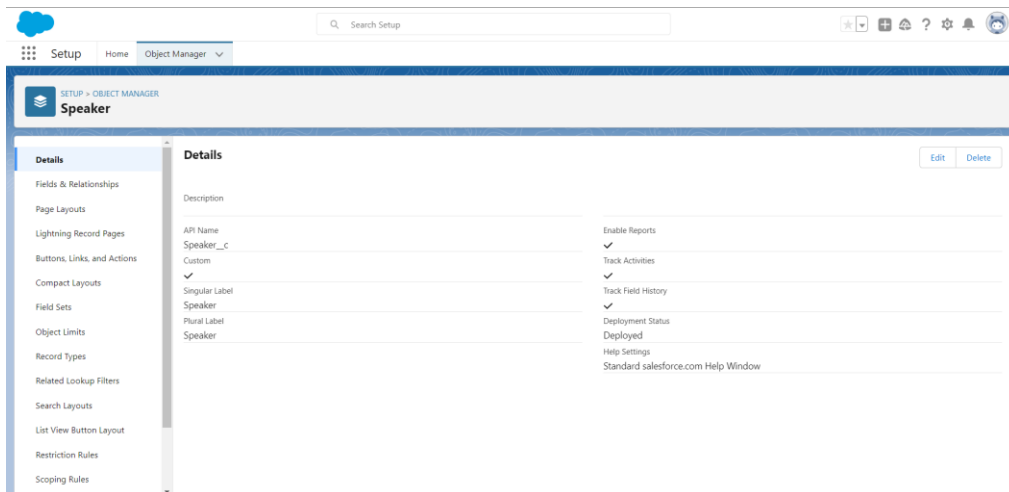
**Fig 3.2: Location**



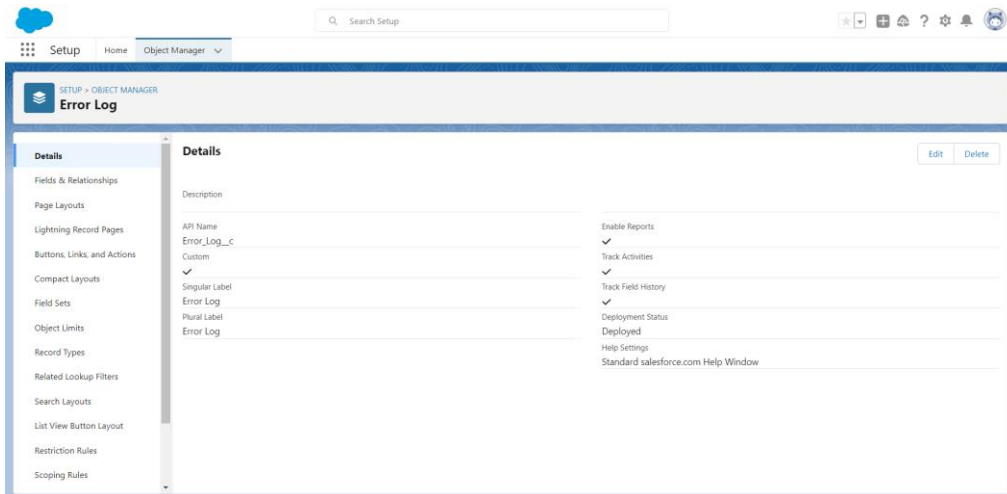
**Fig 3.3: Events**



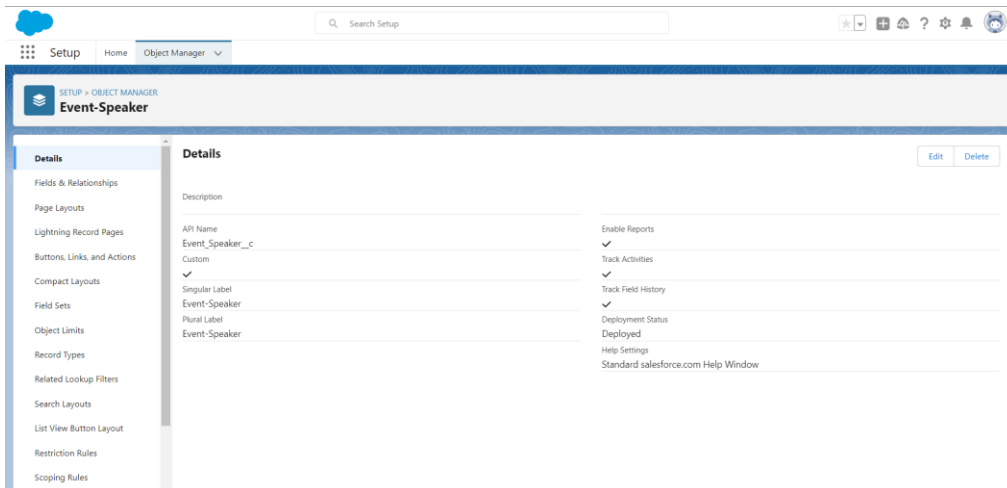
**Fig 3.4: Attendees**



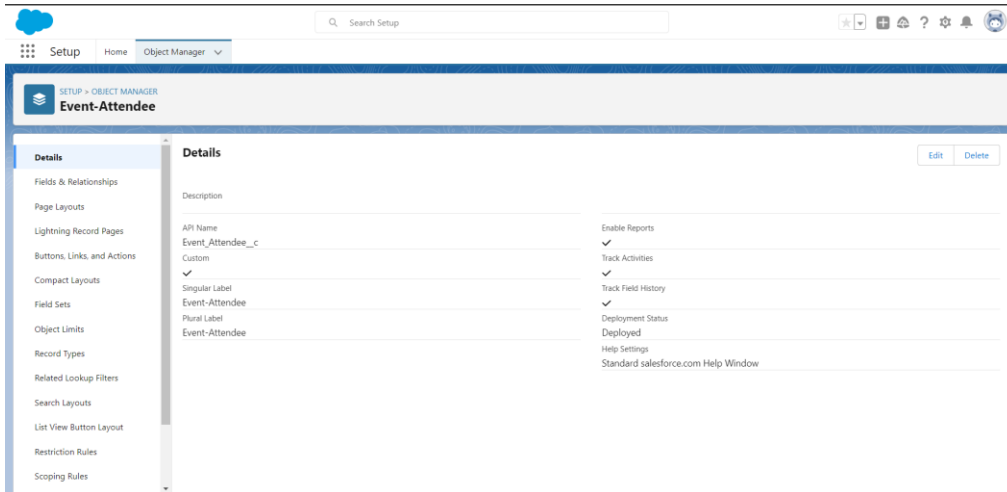
**Fig 3.5: Speaker**



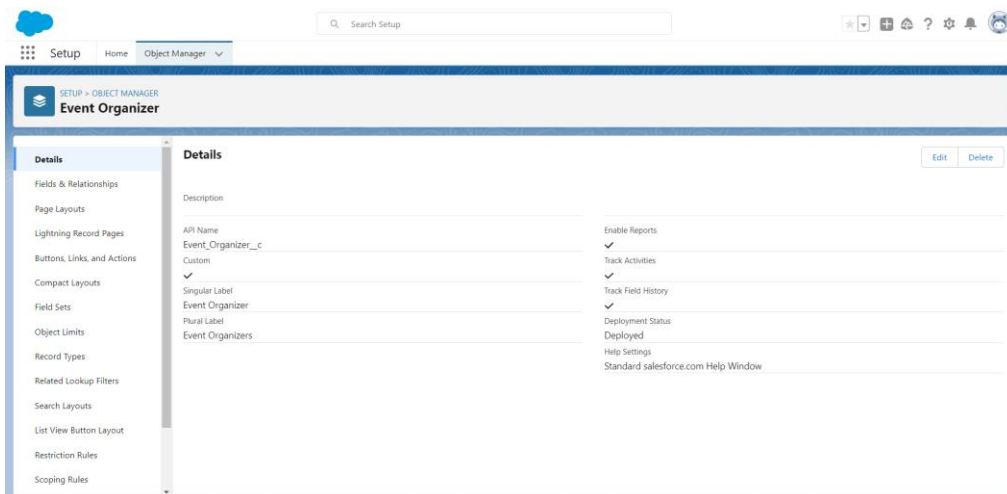
**Fig 3.6: Error log**



**Fig 3.7: Event -Speaker**



**Fig 3.8: Event-Attendee**



**Fig 3.9: Event Organizer**

Objects consist of various components that define their structure and behavior:

**Fields:** Fields represent individual data elements that can be stored within an object. They define the type of data that can be stored, such as text, number, picklist, or date. Fields can also have additional properties like being required, unique, or formula-based.

**Records:** Records are instances or entries of an object that store specific sets of data. Each record contains values for the fields defined within the object and represents a distinct unit of information.

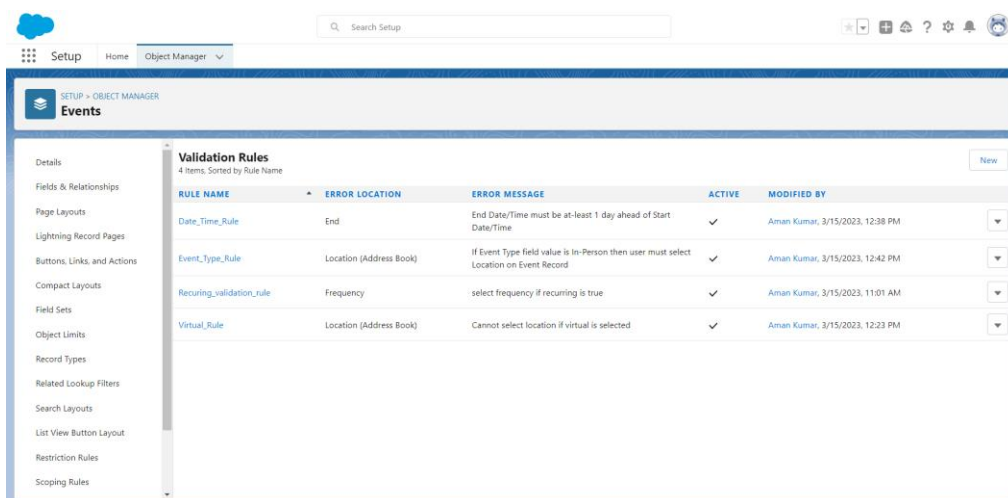
**Relationships:** Relationships define how objects are related to each other. Salesforce supports different types of relationships, including lookup relationships, master-detail relationships, and many-to-many relationships. Relationships enable the establishment of connections and navigation between objects.

**Page Layouts:** Page layouts determine the visual arrangement and organization of fields, related lists, and sections within an object's record detail page. They provide the user interface for viewing and editing records, allowing users to interact with the data effectively.

**Validation Rules:** Validation rules enforce data quality and integrity by specifying criteria that data must meet before it can be saved. These rules help maintain consistent and accurate data entry by validating the values entered into an object's fields.

### Validation Rule on Events:

- Data\_Time\_Rule
- Event\_Type\_Rule
- Recurring\_validation\_Rule
- Virtual\_Rule

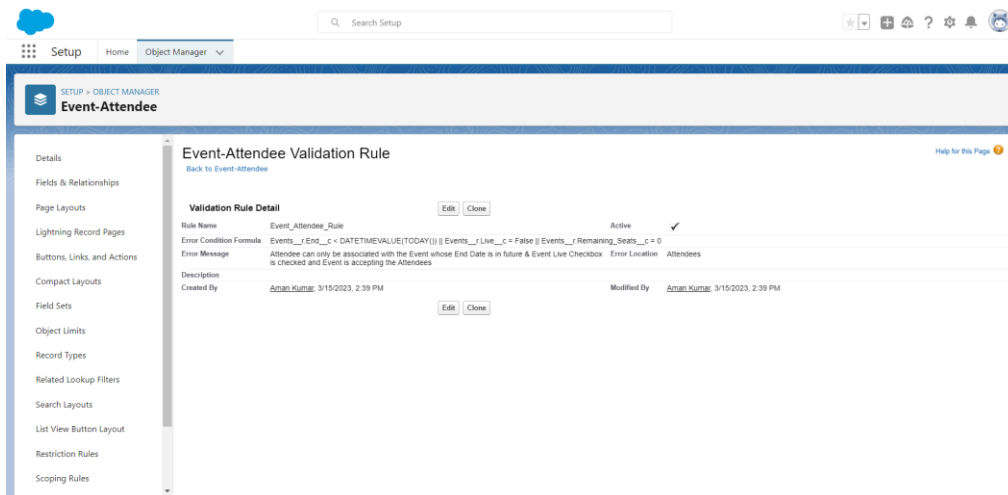


The screenshot shows the Salesforce Setup interface for the 'Events' object. The 'Validation Rules' section is active, displaying a table with 4 items. The table columns are: RULE NAME, ERROR LOCATION, ERROR MESSAGE, ACTIVE, and MODIFIED BY. The rules listed are: Date\_Time\_Rule, Event\_Type\_Rule, Recurring\_validation\_rule, and Virtual\_Rule.

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
Date_Time_Rule	End	End Date/Time must be at-least 1 day ahead of Start Date/Time	✓	Aman Kumar, 3/15/2023, 12:38 PM
Event_Type_Rule	Location (Address Book)	If Event Type field value is In-Person then user must select Location on Event Record	✓	Aman Kumar, 3/15/2023, 12:42 PM
Recurring_validation_rule	Frequency	select frequency if recurring is true	✓	Aman Kumar, 3/15/2023, 11:01 AM
Virtual_Rule	Location (Address Book)	Cannot select location if virtual is selected	✓	Aman Kumar, 3/15/2023, 12:23 PM

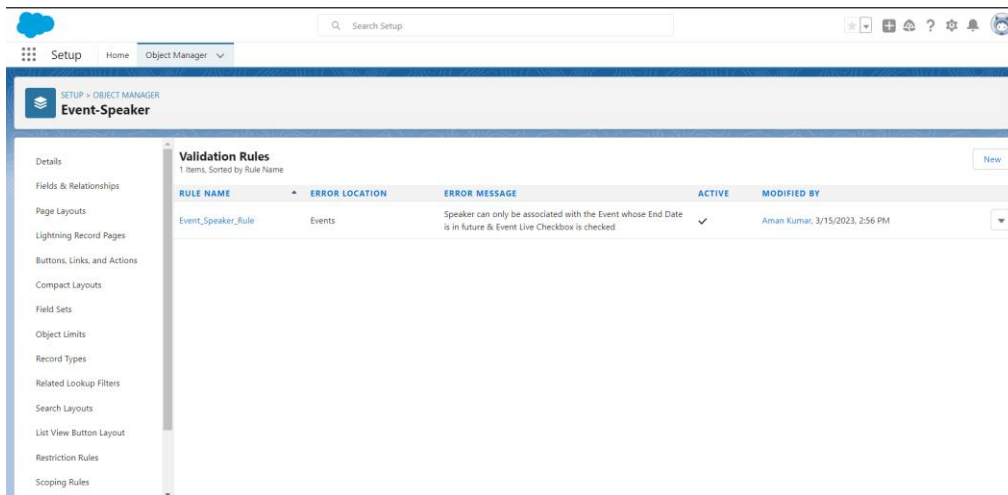
**Fig 3.10: Validation Rules**

## Event\_Attendee\_Rule



**Fig 3.11: Validation Rule on Event Attendee**

## Event\_Speaker\_Rule



**Fig 3.12: Validation Rule on Event-Speaker**

**Object Permissions:** Object permissions control user access to an object and its records. Administrators can manage permissions to determine who can view, create, edit, or delete records of a specific object, ensuring data security and governing access to sensitive information.

By leveraging standard objects and creating custom objects, Salesforce users can store, manage, and analyze data in a way that aligns with their organization's unique business processes and workflows.



## Duplicate Rule:

In Salesforce, a duplicate rule is a feature that prevents the creation of duplicate records within the system, ensuring data integrity. By comparing new record data with existing records, duplicate rules identify potential duplicates and take appropriate actions to avoid duplication.

Duplicate rules are defined based on specific criteria or matching rules. When a user tries to create a new record, the duplicate rule evaluates the entered data against existing records to identify any potential duplicates. If a match is found based on the defined criteria, the rule can display an alert, block record creation, or allow the user to proceed with caution.

Duplicate rules can be applied to both standard and custom objects in Salesforce. The criteria for identifying duplicates can be based on specific fields or combinations of fields. Administrators have the flexibility to define matching logic, such as exact matches, fuzzy matching, or customized algorithms, based on their organization's requirements.

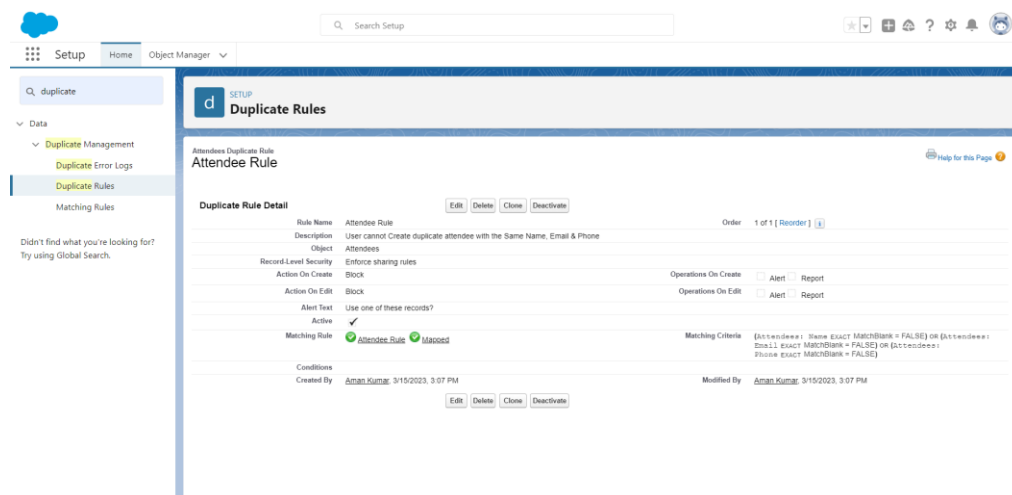
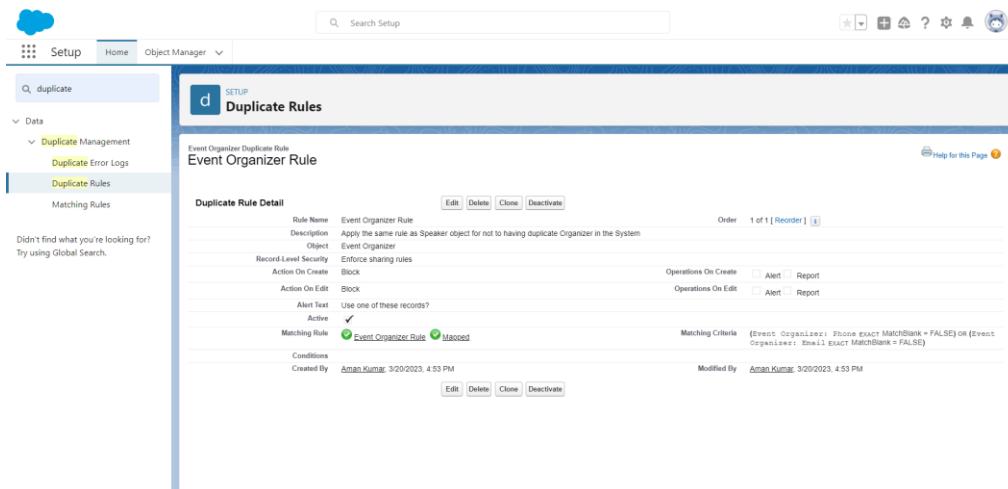
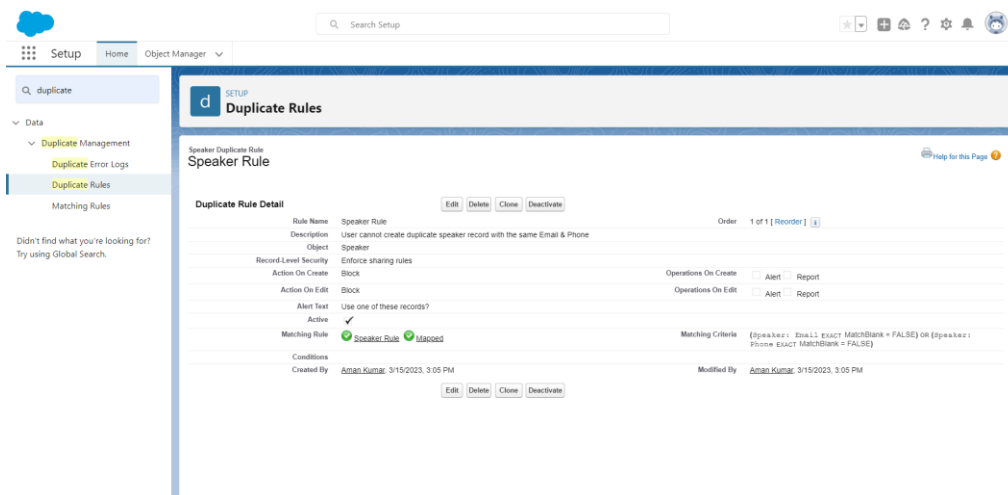


Fig 3.13: Attendee Rule



**Fig 3.14: Event Organizer Rule**



**Fig 3.15: Speaker Rule**

### **Organization Wide Default (OWD)**

In Salesforce, OWD refers to the default level of access and visibility settings applied to records within an organization. It determines the baseline access permissions for objects and records in Salesforce.

OWD settings define how records are shared among users and groups within the organization. There are three main OWD options in Salesforce:

**Private:** With the Private OWD setting, only the record owner and users above them in the role hierarchy have access to the record. Other users, even within the same organization, cannot view or edit the record unless granted explicit access.

**Public Read Only:** In this OWD setting, all users within the organization can view the records, but they cannot edit or delete them. Only the record owner and users with higher roles can modify the records.

**Public Read/Write:** With the Public Read/Write OWD setting, all users within the organization have read and write access to the records. They can view, edit, and delete the records, regardless of their role or position in the organization's hierarchy.

OWD settings can be customized on a per-object basis, allowing organizations to define different levels of access and visibility for different types of records. Additionally, Salesforce provides additional sharing mechanisms like sharing rules, manual sharing, and criteria-based sharing rules to further refine record access and visibility.

	Profile		
Object Name	<i>Event Manager</i>	Speaker	<i>Attendee</i>
<i>Event</i>	CRED	R	R
<i>Event - Organizer</i>	CRE	R	R
<i>Speaker</i>	CRE	CRED	R
<i>Attendee</i>	R	X	CRE
<i>Location</i>	CRED	R	RCE
<i>Event - Speaker</i>	CRED	RCE	R
<i>Event - Attendees</i>	CRED	X	RC

**Fig 3.16: Object Permission Set-up**

## **Development:**

### **Apex Class:**

An Apex class in Salesforce is a fundamental component used to create custom business logic and functionality within the platform. It resembles classes in object-oriented programming languages. Apex serves as the programming language specific to Salesforce, and developers utilize Apex classes to define customized behaviors and actions. Apex classes are employed to implement business processes, manipulate data, interact with the Salesforce database, and execute calculations or complex operations. They enable developers to extend the platform's standard functionality and tailor it to meet specific business requirements. Comprising methods, variables, and properties, Apex classes dictate the behavior and functionality of the class. Methods encompass blocks of code that execute specific actions, such as querying data, updating records, or performing calculations. Variables and properties store data values used within the class, accommodating information such as text, numbers, or dates. Apex classes can be invoked through various means within the Salesforce platform. They are applicable in triggers, which execute code before or after specific events like record creation, updates, or deletions. Additionally, Apex classes are utilized in custom buttons or links on Salesforce pages, enabling specific actions upon user interaction.

Additionally, Visualforce sites commonly make use of Apex classes. These pages employ a combination of HTML, CSS, and Apex code to make it easier to create unique user interfaces. These customised pages' underlying logic and data processing are provided by apex classes.

Additionally, Apex classes enable communication with other systems via outgoing messages, API connections, and web service calls. In the end, Apex classes act as the building blocks for implementing unique business logic and functionality in Salesforce. They enable programmers to alter the platform, automate procedures, work with data, and combine it with external systems to meet certain organisational requirements.

```

public with sharing class DeleteEventRecords implements Database.Batchable<SObject> {
    public Database.QueryLocator start(Database.BatchableContext batchContext){
        String query = 'Select id from Events__c where Live__c = true AND End__c < LAST_N_MONTHS:2';
        return Database.getQueryLocator(query);
    }

    public void execute(Database.BatchableContext batchContext, List<Events__c> listEvt){
        delete listEvt;
    }

    public void finish(Database.BatchableContext batchContext){
        Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();

        mail.setSubject('Batch has been processed');
        List<String> toAddress = new List<String>();
        toAddress.add('aman.kumar@mirketa.com');
        mail.setToAddresses( toAddress );
        mail.setSenderDisplayName('Organizer');

        mail.setHtmlBody('The records have been deleted.');
```

**Fig 3.17: Database Batchable Class**

```

public with sharing class ErrorLogRecordingInsert {
    public static void insertErrorLog(String logDetails, String processName ) {
        Error_log__c errorlog= new Error_log__c();
        errorlog.Log_DateTime__c=DateTime.now();
        errorlog.Log_Details__c=logDetails;
        errorlog.Process_Name__c=processName;
        insert errorlog;
    }
}

```

**Fig 3.18: Error Log Record Class**

### Triggers:

In Salesforce, a trigger is a section of Apex code that executes automatically before or after specific database events, such as the addition, modification, or deletion of records. Developers can construct custom logic and actions that are carried out in response to these events using triggers.

In Salesforce, some objects, such as Accounts, Contacts, or Custom Objects, are linked to particular triggers. The trigger is activated when a record of the connected object experiences the given event. This functionality enables the Salesforce platform's basic capabilities to be extended and customised to fit particular business requirements.

The programming language Apex, which is exclusive to Salesforce, is used to create triggers. They are comprised of event handlers that include code blocks outlining the actions to be taken when the related event happens, and they are saved as Apex classes inside the organisation.

There are two categories of triggers in Salesforce:

**Triggers that run before an event:** These triggers run ahead of the event that set them off. Before the record is stored to the database, they enable developers to carry out operations like data validation, enrichment, or alteration. Prior to doing additional database operations or changing the values of the record, triggers.

**After Triggers:** These triggers go into action following the event that set them off. They provide users access to the data that has been recorded and are frequently used for tasks like updating related records, delivering alerts, or triggering external systems depending on record changes. Within the Salesforce platform, triggers are useful for enforcing business rules, guaranteeing data integrity, automating procedures, and building complicated workflows. They make it easier for custom code to be integrated with database processes, assuring the precise and consistent execution of certain actions in response to data changes.

```

public with sharing class EventAttendeeTriggerHandler {
    public static void AttendeeEmailSend(List<Event_Attendee__c> eventAttendeeNewRecords) {
        List<Messaging.SingleEmailMessage> emails = new List<Messaging.SingleEmailMessage>();

        Set<Id> eventIds = new Set<Id>();
        Set<Id> attendeeIds = new Set<Id>();
        for (Event_Attendee__c eventAttendeeNewRecord : eventAttendeeNewRecords) {

            eventIds.add(eventAttendeeNewRecord.Events__c);
            attendeeIds.add(eventAttendeeNewRecord.Attendees__c);
        }
        //fetching values of event
        Map<Id, Events__c> eventMap = new Map<Id, Events__c>([
            SELECT Name__c, Start__c, Location_Address_Book__r.Name, Organizer__r.Name
            FROM Events__c
            WHERE Id IN :eventIds
        ]);
        //creating a map for attendee event
        Map<Id, Attendees__c> attendeeMap = new Map<Id, Attendees__c>([
            SELECT Name, Email__c
            FROM Attendees__c
            WHERE Id IN :attendeeIds]);

        for (Event_Attendee__c eventAttendeeNewRecord : eventAttendeeNewRecords) {
            if (eventAttendeeNewRecord.Events__c == null && eventAttendeeNewRecord.Attendees__c == null) {
                continue;
            }

            Events__c event = eventMap.get(eventAttendeeNewRecord.Events__c);
            Attendees__c attendee = attendeeMap.get(eventAttendeeNewRecord.Attendees__c);

            Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
            email.setToAddresses(new String[]{attendee.Email__c});
            email.setSubject('Pass for the "' + event.Name__c + '"');
            email.setPlainTextBody('Dear ' + attendee.Name + ',\n\nThank you for registering for "'
                + event.Name__c + '" which will be organized on '
                + event.Start__c.format('dd/mm/yyyy') + ' and will be held in '
                + event.Location_Address_Book__r.Name + '. We are excited to have you, see you at
                the event.\n\nThanks,\n'
                + event.Organizer__r.Name);

            emails.add(email);
        }

        try {
            Messaging.sendEmail(emails);
        } catch (Exception e) {
            ErrorLogRecordingInsert.insertErrorLog(e.getMessage(), 'Error sending email');
        }
    }
}

```

**Fig 3.19: Event Attendee Trigger class**

```

trigger EventAttendeeTrigger on Event_Attendee__c (after insert){
    EventAttendeeTriggerHandler.AttendeeEmailSend(trigger.new);
}

```

**Fig 3.20: Event Attendee Trigger**

```

trigger EventSpeakerTrigger on Event_Speaker__c (before insert, before update) {
    Set<Id> eventId = new Set<Id>();
    Set<Id> speakerId = new Set<Id>();
    for(Event_Speaker__c evtSpkr: Trigger.new){
        eventId.add(evtSpkr.Events__c);
        speakerId.add(evtSpkr.Speaker__c);
    }
    Map<Id,Events__c> evtDateTime= new Map<Id, Events__c>([Select id, Start__c, End__c from Events__c where id in :eventId]);
    List<Event_Speaker__c> spkr = [Select id, Events__c,Speaker__c,Events__r.Start__c, Events__r.End__c from Event_Speaker__c
    where Speaker__c in :speakerId];
    for(Event_Speaker__c evtSpkr: Trigger.new){
        Events__c evtTime = evtDateTime.get(evtSpkr.Events__c);
        Datetime evtTimeStart = evtTime.Start__c;
        Datetime evtTimeEnd = evtTime.End__c;
        for(Event_Speaker__c checkEvtSpkr: spkr){
            if(evtSpkr.Speaker__c == checkEvtSpkr.Speaker__c && ((evtTimeStart <= checkEvtSpkr.Events__r.Start__c &&
            evtTimeEnd >= checkEvtSpkr.Events__r.End__c) ||(evtTimeStart >= checkEvtSpkr.Events__r.Start__c && evtTimeStart <=
            checkEvtSpkr.Events__r.End__c) ||(evtTimeEnd >= checkEvtSpkr.Events__r.Start__c && evtTimeEnd <= checkEvtSpkr.
            Events__r.End__c)))
            {
                evtSpkr.Speaker__c.addError('Already booked');
                evtSpkr.addError('Already booked');
            }
        }
    }
}

```

**Fig 3.21: Event Speaker Trigger**

### Visualforce Pages:

Visualforce is a framework within Salesforce that empowers developers to build personalized user interfaces for their applications. These interfaces, known as Visualforce pages, are constructed using a blend of HTML, CSS, and Apex code.

A Visualforce page defines the arrangement and structure of the user interface, encompassing the positioning of fields, buttons, and other elements. It also allows for the integration of dynamic content and data retrieved from the Salesforce database by leveraging Apex controllers or extensions.

Visualforce pages can be tailored to match the organization's branding and style, ensuring a consistent and cohesive user experience. They support a variety of input components, including fields for data entry, checkboxes, dropdown menus, and buttons, enabling users to interact seamlessly with the displayed information.

Developers can embed business logic and data manipulation functionalities within Visualforce pages by utilizing Apex controllers. These controllers serve as the backend logic and data access layer, facilitating calculations, queries, and updates as required.



Visualforce pages can be accessed via web browsers or incorporated into Salesforce as tabs, related lists, or custom buttons. They serve a multitude of purposes, such as creating customized data entry forms, generating reports and dashboards, constructing bespoke interfaces for specific business processes, or integrating external systems.

Through Visualforce, developers gain the flexibility to craft personalized and interactive user interfaces that align with the unique requirements of their organization. This framework enhances the overall user experience within the Salesforce platform, fostering greater engagement and productivity.

Visualforce pages in Apex provide a powerful way to create user interfaces and display data from your Salesforce application. Here are some ways in which Visualforce pages are helpful in Apex development:

**User Interface Development:** Visualforce allows you to create custom user interfaces for your Salesforce application. You can design pages with components such as forms, tables, buttons, and charts to interact with users and display data.

**Data Display and Manipulation:** With Visualforce, you can query data from Salesforce objects and display it in tables, lists, or custom layouts. You can also perform data manipulation operations like inserting, updating, and deleting records using Visualforce controllers and standard or custom Apex classes.

**Custom Business Logic:** Visualforce pages can be associated with Apex controllers, which are classes that provide the logic and data for the page. You can use Apex controllers to implement custom business processes, perform calculations, validate user input, and interact with external systems.

**Integration with Apex Code:** Visualforce pages can invoke Apex methods and controllers to perform complex business operations and data processing. You can leverage the power of Apex code to handle data manipulation, perform calculations, and interact with external systems.

```
public with sharing class EventRegistrationController {
    public Event_Organizer__c evtOrganizer {get;set;}
    public Events__c evt {get;set;}
    public Location__c loc {get;set;}

    public EventRegistrationController ()
    {
        evtOrganizer = new Event_Organizer__c();
        evt = new Events__c();
        loc = new Location__c();
    }
    public PageReference save()
    {
        // insert con;
        if(evt.Event_Type__c != 'Virtual'){
            insert loc;
            evtOrganizer.Address__c = loc.Id;

            insert evtOrganizer;
            evt.Organizer__c = evtOrganizer.Id;
            evt.Location_Address_Book__c = loc.Id;
            insert evt;}
        else{
            insert evtOrganizer;
            evt.Organizer__c = evtOrganizer.Id;
            // evt.Location_Address_Book__c = loc.Id;
            insert evt;
        }
        return new PageReference('/apex/EventRegistrationController').setRedirect(true);
    }
}
```

**Fig 3.22: Event Registration Controller Class for Visualforce Page**

```

public with sharing class AttendeeRegistrationController {
    public Attendees__c att {get;set;}
    public Event_Attendee__c evtAttendee{get;set;}
    public Location__c loc {get;set;}
    public String mstrId;
    public AttendeeRegistrationController() {
        mstrId = ApexPages.currentPage().getParameters().get('param1');
        att= new Attendees__c();
        evtAttendee = new Event_Attendee__c();
        loc= new Location__c ();
        evtAttendee.Events__c = mstrId;
    }
    public PageReference save(){
        insert loc;
        att.Location_Address_Book__c=loc.Id;
        insert att;
        evtAttendee.Attendees__c= att.Id;
        insert evtAttendee;
        return new PageReference('/apex/AttendeeRegistrationController').setRedirect(true);
    }
}

```

**Fig 3.23: Event Attendee Controller Class for Visualforce Page**

```

<apex:page controller="EventRegistrationController">
    <apex:form>
        <apex:pageBlock title="Event Registration">
            <!-- <apex:pageBlockButtons> -->

            <!-- </apex:pageBlockButtons> -->
            <apex:pageBlockSection title="Event organizer" collapsible="true" columns="2" >
                <apex:inputField value="{!evtOrganizer.Name}" />
                <apex:inputField value="{!evtOrganizer.Phone__c}" />
                <apex:inputField value="{!evtOrganizer.Email__c}" />
            </apex:pageBlockSection>

            <apex:pageBlockSection title="Event" collapsible="true" columns="2">
                <apex:inputField value="{!evt.Name__c}" />
                <apex:inputField value="{!evt.Event_Type__c}" />
                <apex:inputField value="{!evt.Max_Seats__c}" />
                <apex:inputField value="{!evt.Start__c}" />
                <apex:inputField value="{!evt.End__c}" />
                <apex:inputField value="{!evt.Live__c}" />
                <apex:inputField value="{!evt.Recurring__c}" />
                <apex:inputField value="{!evt.Frequency__c}" />
            </apex:pageBlockSection>

            <apex:pageBlockSection title="Location" collapsible="true" columns="2">
                <apex:inputField value="{!loc.Name}" />
                <apex:inputField value="{!loc.Landmark__c}" />
                <apex:inputField value="{!loc.Street__c}" />
                <apex:inputField value="{!loc.City__c}" />
                <apex:inputField value="{!loc.Postal_Code__c}" />
                <apex:inputField value="{!loc.Country__c}" />
            </apex:pageBlockSection>
            <div align="center" draggable="false" >
                <apex:commandButton value="Save" action="{!save}" />
            </div>
        </apex:pageBlock>
    </apex:form>
</apex:page>

```

**Fig 3.24: Event Registration Form**

```

<apex:page controller="AttendeeRegistrationController">
  <apex:form>
    <apex:pageBlock title='Attendee Registration ' >
      <apex:pageBlockSection title="Attendee Detail" collapsible="true" columns="2">
        <apex:inputField value="{!att.Name}"/>
        <apex:inputField value="{!att.Email__c}"/>
        <apex:inputField value="{!att.Phone__c}"/>
        <apex:inputField value="{!evtAttendee.Events__c}" rendered="{!evtAttendee.Events__c==null}"/>
        <apex:outputField value="{!evtAttendee.Events__c}" rendered="{!evtAttendee.Events__c!=null}"/>
      </apex:pageBlockSection>
      <apex:pageBlockSection title="Location" collapsible="true" columns="2">
        <apex:inputField value="{!loc.Name}"/>
        <apex:inputField value="{!loc.Street__c}"/>
        <apex:inputField value="{!loc.City__c}"/>
        <apex:inputField value="{!loc.State__c}"/>
        <apex:inputField value="{!loc.Country__c}"/>
        <apex:inputField value="{!loc.Postal_Code__c}"/>
      </apex:pageBlockSection>
      <div align="center" draggable="false" >
        <apex:commandButton value="Save" action="{!save}"/>
      </div>
    </apex:pageBlock>
  </apex:form>
</apex:page>

```

**Fig 3.25: Attendee Registration Form**

```

public with sharing class SpeakerRegistrationController {

  public Speaker__c spk{get;set;}
  public Event_Speaker__c espk{get;set;}
  //public Events__c evt{get;set;}
  public String mstrId;

  public SpeakerRegistrationController() {
    mstrId = ApexPages.currentPage().getParameters().get('param1');
    spk=new Speaker__c();
    espk=new Event_Speaker__c();
    //evt=new Events__c();
    espk.Events__c=mstrId;
  }
  public PageReference save()
  {
    insert spk;
    espk.Speaker__c=spk.ID;

    insert espk;

    return new PageReference('/apex/SpeakerRegistrationController').setRedirect(true);
  }
}

```

**Fig 3.26: Speaker Registration Controller Class for Visualforce Page**

## Chapter 4

### EXPERIMENTS & RESULT ANALYSIS

The project's experiments are concentrated on creating validation rules, duplication rules, profiles, roles, sharing rules, and triggers on various system objects. A reusable Apex class, an Apex batch, and forms for event registration, speaker registration, and audience registration are all part of the experiments. In order to guarantee that the implemented code is verified and validated for at least 90% code coverage, a unit test is also built.

The Event, Event Attendee, and Event Speaker objects were given the necessary data integrity checks through the establishment of the validation rules. For example, if the "Recurring?" checkbox is checked, the Frequency field must be filled, and conversely, if the checkbox is unchecked, the Frequency field should not be selectable. Another example is that when the Event Type field is set to "Virtual," the validation rule should prevent users from selecting a Location on the Event record. These validation rules help to ensure that the data entered into the system is correct and consistent.

To avoid duplicate records in the Speaker, Attendee, and Organiser objects, the duplication rules were put up. To avoid multiple Organisers in the system, for instance, the same duplicate rule as the Speaker object was used. These rules help to prevent the system from being cluttered with redundant and confusing data.

The profiles, roles, and sharing rules were set up to ensure proper access control in the system. The system has three profiles (Event Organizer, Event Attendee, and Speaker) created by cloning from the Standard User profile. Three roles (Organizer, Attendee, and Speaker) were created, with all roles reporting to the CEO. A sharing rule was created to share Speaker and Attendee records with the Organizer role, granting read/edit permissions.

The Trigger Development experiments were focused on developing triggers on the Event Speaker and Event Attendee objects. The Event Speaker object trigger was designed to throw an error if the Speaker Selected on Event - Speaker Record already has an Event against his name. The Event Attendee object trigger was designed to send an email to Attendee saying that registration has been confirmed.

The Apex Batch experiment was focused on deleting all the event records which are more than 2 months old and have been organized. The finish method of the batch apex sends an email to the admin saying that execution of the batch has been processed.

The unit test experiment was focused on creating a unit test for the implemented code. The unit test ensures that the implemented code is tested and validated for at least 90% code coverage. The unit test uses @TestSetup annotation, Test.startTest, Test.stopTest, and Asserts methods. The test also includes negative test cases to ensure that the code works correctly even when invalid data is entered into the system.

The event registration, speaker registration, and attendee registration forms were created to allow users to easily register for events, speakers, and attendees. The forms are designed to ensure that the required data is collected from the user and that the data is entered correctly into the system. Upon save, a record for event organizer, event and location should be created. Upon save, a record for speaker, event-speaker should be created. The form for attendee registration includes a lookup field to select an event, ensuring that attendees are registered for the correct event.

## Output at various stages:

### Validation Rule:

Live?

Recurring?   
[View all dependencies](#)

Event Type

Frequency   
select frequency if recurring is true  
[View all dependencies](#)

Location (Address Book)

Organizer

**⊘ We hit a snag.**  
Review the following fields  
• [Frequency](#)

**Fig 4.1: Recuring\_validation\_rule**

Event Type

Frequency   
[View all dependencies](#)



Location (Address Book)  
  
Cannot select location if virtual is selected


Organizer



**⊘ We hit a snag.**  
Review the following fields  
• [Location \(Address Book\)](#)

**Fig 4.2: Virtual\_Rule**

**Start** ⓘ

Date: 4/14/2023  Time: 12:00 PM 

**End** ⓘ 

Date: 4/12/2023  Time: 12:00 PM 

End Date/Time must be at-least 1 day ahead of Start Date/Time

Max Seats ⓘ

Live? ⓘ


Recurring? ⓘ  [View all dependencies](#)

**Event Type** ⓘ

Virtual


**Frequency** ⓘ

Weekly


**We hit a snag.** 


Review the following fields

- [Location \(Address Book\)](#)
- End




**Fig 4.3: date time validation rule**

**Event Type** ⓘ 


In-Person 

**Frequency** ⓘ

Weekly 

[View all dependencies](#)


Location (Address Book)

Search Locations... 

If Event Type field value is In-Person then user must select Location on Event Record


Organizer

Search Event

**We hit a snag.** 

Review the following fields

- [Location \(Address Book\)](#)



**Fig 4.4: Event\_Type\_Rule**



## Duplicate Rule:

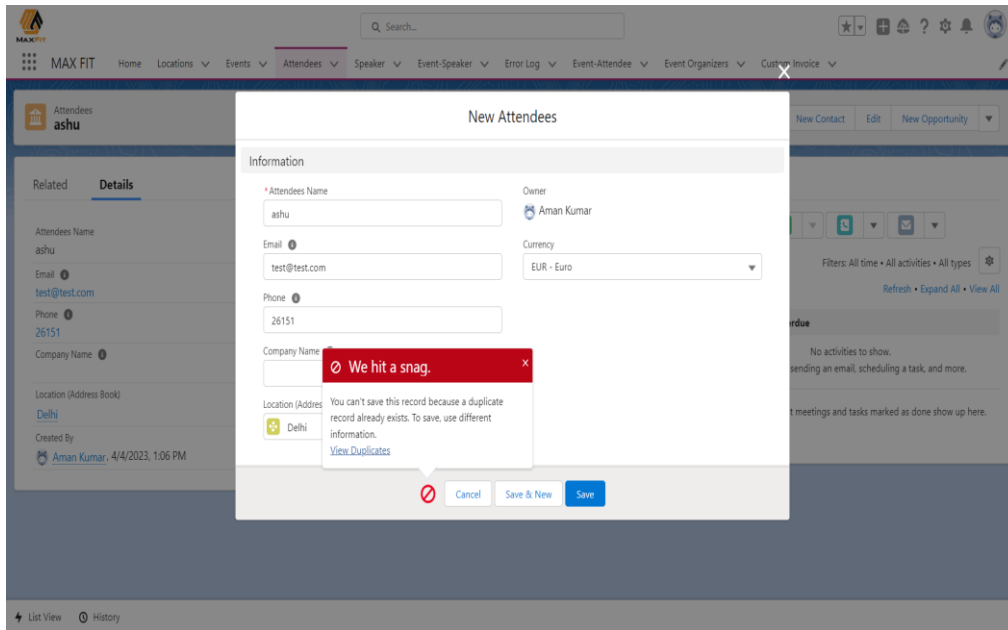


Fig 4.5: Attendee Rule

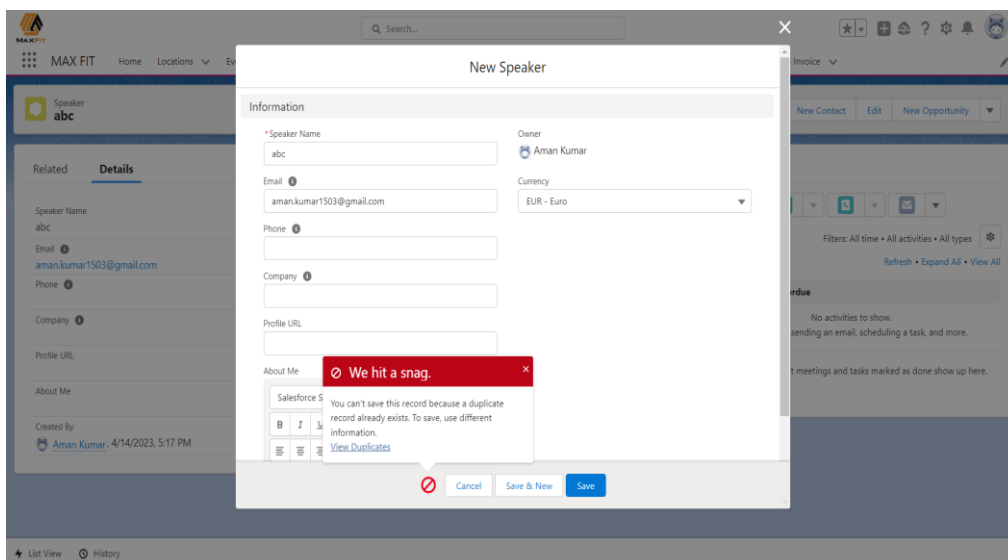
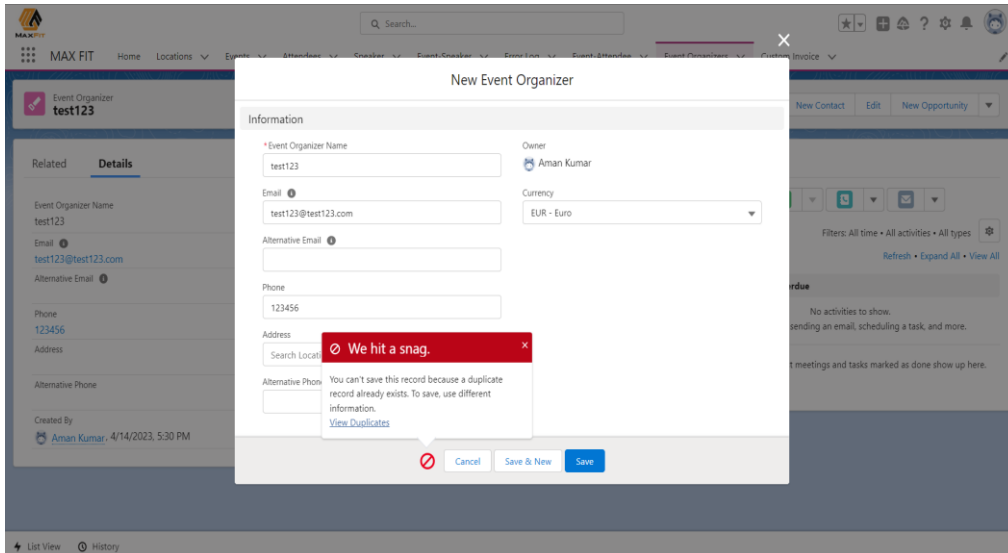
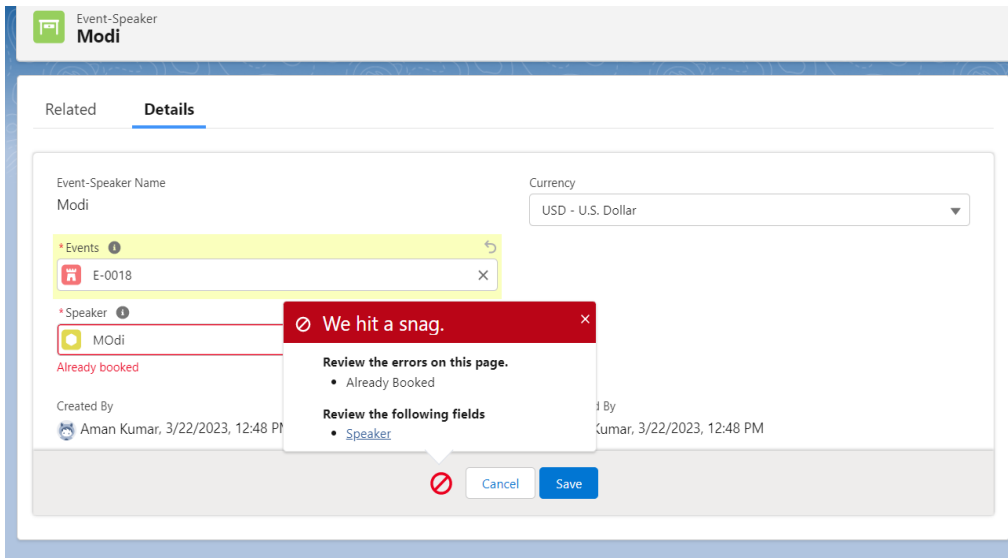


Fig 4.6: Speaker Rule



**Fig 4.7: event organizer rule**

## Trigger



**Fig 4.8: EventSpeakerTrigger**

Pass for the "test2" Inbox x



**Aman Kumar** via 0tvb6tjr03nmwbbwvhg4m5s.2w-qvy6peav.ap16.bnc.salesforce.com  
to me ▾

12:17 AM (6 minutes ago)

Dear aman,

Thank you for registering for "test2" which will be organized on 05/37/2023 and will be held in null. We are excited to have you, see you at the event.

Thanks,  
test2

**Fig 4.9: EventAttendeeTrigger**

## Visualforce Pages:

The screenshot displays the 'Event Registration' page with three main sections: 'Event organizer', 'Event', and 'Location'. Each section contains several input fields and dropdown menus. The 'Event organizer' section includes fields for Name, Email, and Phone. The 'Event' section includes fields for Name, Max Seats, End date, Event Type, Start date, Live?, and Frequency. The 'Location' section includes fields for Location Name, Street, Postal Code, Landmark, City, and Country. A 'Save' button is located at the bottom center of the form. The browser's address bar at the bottom shows the URL 'EventRegistrationController'.

**Fig 4.10: Event Registration Controller Page**

**Speaker Registration**

▼ Speaker Registration

Speaker Name

Phone

Events

Email

Company

SpeakerRegistrationController SpeakerRegistrationController

**Fig 4.11: Speaker Registration Controller Page**

**Attendee Registration**

▼ Attendee Detail

Attendees Name

Phone

Events

Email

▼ Location

Location Name

City

Country

Street

State

Postal Code

AttendeeRegistrationController AttendeeRegistrationController

**Fig 4.12: Attendee Registration Controller Page**

## Chapter 5

### CONCLUSIONS

#### 5.1 Conclusions

In conclusion, the project aimed to improve the functionality, data integrity, and access control of an event management system through several milestones.

In Milestone 1, validation rules were set up to enforce various requirements on the Event object, Event Attendee object, and Event Speaker object. These rules ensured that the system validated the data entered and prevented inconsistencies and errors.

Milestone 1.2 focused on setting up duplicate rules to avoid the creation of duplicate records for the Speaker, Attendee, and Organizer objects. This step helped maintain data integrity and avoid confusion.

Profile, user, organization-wide default (OWD), and role setups were performed in Milestone 1.3 to establish proper access control within the system. Cloning from the Standard User profile, three profiles (Event Organizer, Event Attendee, and Speaker) were created. Users were also set up for testing purposes, and roles were defined, with a hierarchical structure.

Sharing rules were implemented to ensure that Speaker and Attendee records were shared with the Organizer role, providing the necessary access permissions. This setup improved collaboration and information sharing within the system.

Milestone 2 involved the development of a reusable Apex Class that contained a method to insert Error log Object records dynamically. This class facilitated error handling and logging, capturing relevant details for effective debugging and monitoring.

Milestone 3 focused on the development of a Trigger on the Event-Speaker object to prevent duplicate bookings. The trigger checked if a speaker was already associated with an event and rejected duplicate bookings, ensuring that each speaker was associated with only one event at a time.

In Milestone 4, an Apex Trigger was developed on the Event Attendee object to send confirmation emails to attendees when a new record was created. The trigger automatically generated personalized confirmation emails with event details, enhancing the attendee experience.

Milestone 5 involved the development of an Apex Batch to delete event records that were more than 2 months old and had already occurred. This batch process helped maintain data cleanliness and system performance. An email notification was sent to the admin upon completion of the batch execution.

In Milestone 6, comprehensive unit tests were developed to achieve at least 90% code coverage. Various testing techniques and best practices were applied to ensure the functionality and correctness of the implemented code.

Milestone 7 focused on creating a form for event registration with collapsible sections, capturing relevant information about the event organizer, event details, and event location. Upon saving the form, corresponding records were created for the event organizer, event, and location.

Milestone 8 entailed developing a speaker registration form, gathering speaker information, and enabling event selection. Records were made for the speaker and the event-speaker association when the form was saved.

In Milestone 9, an attending registration form was created that collects participant information, including location. The form also offered the option to

choose an event, and after saving, records for the attendee and the relationship between the event and the attendee were generated.

By increasing the procedures for data validation, access control, error handling, email alerts, data cleansing, and user registration, these milestones have overall considerably enhanced the event management system. These improvements make the event management system more effective, dependable, and accessible.

## **5.2 Future Scope**

The event management system has undergone considerable upgrades thanks to the aforementioned initiative. However, there are a number of areas with a future focus that can improve its usability and functionality even further. Future advancements for the project might involve some of the following:

- **Advanced Reporting and Analytics:** Implementing advanced reporting and analytics capabilities can provide valuable insights into event attendance, speaker performance, attendee demographics, and other key metrics. This data can help organizers make informed decisions, identify trends, and optimize future events.
- **Mobile Application:** Developing a mobile application for the event management system can extend its accessibility and convenience. Users can easily access event information, register, and receive notifications directly on their mobile devices, enhancing the overall user experience.

- **Integration with Payment Gateways:** Integrating the system with popular payment gateways enables seamless online event registration and ticketing processes. Attendees can securely make payments, while organizers can track and manage financial transactions more efficiently.
- **Social Media Integration:** It is possible to promote events, encourage sharing, and increase attendee participation by integrating the event management system with social media platforms. Event information can be easily shared by users on their social media sites, boosting event visibility and drawing in a bigger audience.
- **Automated Waitlist Management:** Event planners can effectively manage event waitlists by putting in place an automated system. When a registered attendee cancels, the system can automatically notify and offer the spot to individuals on the waitlist, streamlining the registration process.
- **Enhanced Event Scheduling:** Introducing advanced event scheduling features such as recurring events, multi-day events, and parallel sessions can cater to a wider range of event types and improve flexibility for both organizers and attendees.
- **Integration with External Platforms:** Integrating the event management system with external platforms such as email marketing tools, CRM systems, or collaboration tools can streamline communication, data synchronization, and workflow management, enhancing overall productivity.
- **Feedback and Survey System:** Implementing a feedback and survey system allows organizers to collect valuable insights from attendees, speakers, and organizers. Feedback can be used to improve future events, enhance the attendee experience, and gauge overall satisfaction.



- **Machine Learning and AI Integration:** Leveraging machine learning and artificial intelligence technologies can enable personalized event recommendations, intelligent content suggestions, and automated event planning based on historical data and attendee preferences.

These future scope areas can further elevate the event management system, improving its features, capabilities, and user engagement. It is important to assess the specific requirements, priorities, and feasibility of these enhancements based on the project's goals and stakeholders' needs.

### 5.3 Applications

The application of the above project can be seen in various event management scenarios, where efficient and streamlined processes are required. Here are some examples of how the project can be applied:

- **Conference and Trade Show Management:** The event management system can be utilized to organize and manage conferences and trade shows. It can handle registration, session scheduling, speaker management, attendee tracking, and communication, ensuring a smooth and successful event experience.
- **Corporate Event Management:** Companies often organize internal and external events such as seminars, workshops, and corporate gatherings. The event management system can help in planning and executing these events, managing attendee registrations, sending event updates, and collecting feedback.

- **Educational Event Management:** Educational institutions, including schools and universities, frequently organize events like seminars, career fairs, and student conferences. The project can be applied to efficiently manage event logistics, registrations, speaker coordination, and attendee communication.
- **Community Event Management:** The event management system can be used by community organizations, non-profits, and local governments to manage community events like festivals, fundraisers, and cultural programs. It can facilitate event promotion, volunteer management, ticketing, and overall event coordination.
- **Professional Association Events:** Professional associations and industry groups often organize conferences, workshops, and networking events. The project can be utilized to manage membership registration, session scheduling, speaker management, and event logistics specific to these professional communities.

## References

- [1] Validation Rules:  
[https://developer.salesforce.com/docs/atlas.en-us.validation\\_rules.meta/validation\\_rules/](https://developer.salesforce.com/docs/atlas.en-us.validation_rules.meta/validation_rules/)
  
- [2] Duplicate Management:  
[https://help.salesforce.com/articleView?id=sf.duplicate\\_rules.htm&type=5](https://help.salesforce.com/articleView?id=sf.duplicate_rules.htm&type=5)
  
- [3] Apex Developer Guide:  
<https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/>
  
- [4] Triggers:  
[https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex\\_triggers.htm](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_triggers.htm)
  
- [5] Sending Email Using Apex:  
[https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex\\_classes\\_email\\_outbound.htm](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_classes_email_outbound.htm)
  
- [6] Apex Batch Development:  
<https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/>
  
- [7] Batch Apex:  
[https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex\\_batch.htm](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_batch.htm)
  
- [8] Visualforce Developer Guide:  
[https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages\\_intro.htm](https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages_intro.htm)

am

ORIGINALITY REPORT

1%

SIMILARITY INDEX

0%

INTERNET SOURCES

1%

PUBLICATIONS

0%

STUDENT PAPERS

PRIMARY SOURCES

- 1** Dipanker Jyoti, James A. Hutcherson. "Salesforce Architect's Handbook", Springer Science and Business Media LLC, 2021  
Publication <1%
- 2** Rakesh Gupta. "Salesforce Platform App Builder Certification", Springer Science and Business Media LLC, 2020  
Publication <1%
- 3** Lee Harding, Lee Bayliss. "Salesforce Platform Governance Method", Springer Science and Business Media LLC, 2022  
Publication <1%
- 4** Philip Weinmeister. "Practical Salesforce.com Development Without Code", Springer Science and Business Media LLC, 2015  
Publication <1%
- 5** learnfrenzy.com  
Internet Source <1%
- 6** Rashed A. Chowdhury. "Building a Salesforce-Powered Front Office", Springer Science and Business Media LLC, 2021  
<1%