**Project Title:** Smart Event Management System with Automated Volunteer Assignment

**Problem Statement**

Event management is a fast-growing industry that involves handling multiple stakeholders, including clients, vendors, volunteers, and internal teams. Currently, many event companies face challenges such as manual tracking of bookings, difficulty in coordinating with vendors, lack of transparency in budget approvals, and limited real-time reporting. These inefficiencies lead to errors like double-booking venues, vendor scheduling conflicts, and missed deadlines, ultimately affecting customer satisfaction.

The **Smart Event Management CRM** aims to solve these issues by leveraging Salesforce as a unified platform for event planning, vendor management, client communication, and revenue tracking. By automating processes, enabling real-time collaboration, and providing actionable reports, the solution ensures streamlined operations and better decision-making for event managers and stakeholders.

**Requirement Gathering**

To understand the scope of the project, the following requirements were gathered by analyzing industry practices and interacting with potential stakeholders:

* **Event Tracking**: Ability to manage event details such as venue, date, budget, and expected attendees.
* **Client Management**: Support for client registration, corporate bookings, and personalized services.
* **Vendor Allocation**: Assign vendors for catering, logistics, decoration, and stage setup with availability checks.
* **Approval Workflows**: Automate approvals for high-value events, such as those exceeding a defined budget threshold.
* **Reporting**: Generate reports for revenue, vendor utilization, and event performance.

These requirements ensure that the system addresses real-world event management challenges.

**Stakeholder Analysis**

The system will serve different roles, each with distinct responsibilities:

* **Admin** – Responsible for configuring and managing the Salesforce CRM setup.
* **Event Manager** – Creates and manages events, assigns vendors, and monitors execution.
* **Vendors** – Provide services (catering, decoration, etc.) and need visibility into allocated events.
* **Clients** – Book events, register for participation, and receive confirmations.
* **Finance Team** – Approves high-budget events and monitors payment and revenue flow.

By clearly defining stakeholders, the project ensures that access, permissions, and responsibilities align with business needs.

**Business Process Mapping**

The flow of event execution can be represented as:

1. **Client Request** → Client requests an event (wedding, conference, concert, etc.).
2. **Event Creation** → Event Manager creates the event in Salesforce with details like venue, date, and budget.
3. **Vendor Allocation** → Vendors are assigned to handle specific event services.
4. **Approval** → If the budget exceeds the defined threshold, it goes to Finance Team for approval.
5. **Notifications** → Once approved, notifications are sent to Clients and Vendors.
6. **Execution & Reporting** → The event takes place, and reports are generated for tracking performance and revenue.

This process map highlights how Salesforce automation will reduce manual effort and ensure accuracy.

**Industry-Specific Use Case Analysis**

The event management industry deals with:

* **Time-sensitive operations**: Events must be planned and executed within strict deadlines.
* **Resource dependency**: Vendor availability, venue booking, and budgeting are interlinked.
* **Client expectations**: Clients demand transparency, quick confirmations, and professional execution.

By implementing Salesforce, the system can:

* Avoid scheduling conflicts through automation.
* Provide real-time visibility of vendor and venue availability.
* Generate insights on revenue and event performance, improving business decisions.

**AppExchange Exploration**

During the research phase, existing event management apps on Salesforce AppExchange were explored. While some solutions exist for niche purposes (like ticketing or volunteer management), most are either too generic or too advanced for learning-focused projects.

Hence, **this project builds a custom solution tailored for event companies**, ensuring full control over design while also serving as a strong capstone for Salesforce certification.

**Phase 1: Requirement Gathering**

**Objective:** The aim of this phase is to understand what needs to be built, who will use it, and what exact problems the system must solve.

**Activities:**  
We begin by identifying stakeholders such as administrators (college/organization heads), event organizers (student coordinators or staff), and participants (students, attendees). Through interviews and questionnaires, their expectations and pain points are collected. Organizers, for instance, want an easy way to create and promote events without repetitive manual work. Participants want a simple registration process and timely updates. Administrators want consolidated reporting on event participation and outcomes.

We also document both **functional requirements** (like event creation, participant registration, notifications, dashboards) and **non-functional requirements** (security of participant data, performance under high traffic, mobile-friendly access).

**Deliverables:** A **Requirement Specification Document** containing a list of user stories, example workflows, and prioritization of features.

**Phase 2: Feasibility & Planning**

**Objective:** Assess whether the proposed system is practical, sustainable, and aligned with organizational needs.

**Activities:**

* **Technical feasibility:** Salesforce’s low-code platform, combined with Apex for backend logic and Lightning Web Components for UI, ensures the system can handle the expected scale and features.
* **Operational feasibility:** Since most users (organizers/students) are non-technical, the solution must have an intuitive interface. This is feasible with Salesforce’s drag-and-drop app builder.
* **Economic feasibility:** As we are using a Developer Org for academic purposes, the cost is minimal, making it highly feasible.

A project timeline is prepared, dividing the work into 10 phases with clear milestones. Risks such as delays in training or feature complexity are also documented along with mitigation strategies.

**Deliverables:** Feasibility Report, Risk Register, and Project Plan with Gantt chart.

**Phase 3: High-Level Design**

**Objective:** Define the overall system architecture and how different components will interact.

**Activities:**  
This phase involves building diagrams that capture how the system looks at a macro level. The **system architecture diagram** shows the interaction between Salesforce objects (Event, Registration, Participant), Apex classes for logic, and Lightning Web Components for the interface. **Use Case Diagrams** depict the roles of Admins, Organizers, and Participants. **Data Flow Diagrams** explain how participant data moves from registration to reporting.

Here, we also finalize modules such as “Event Creation,” “Registration,” “Notification,” and “Analytics,” ensuring they align with user needs captured earlier.

**Deliverables:** High-Level Design Document with architecture, use case, and DFD diagrams.

**Phase 4: Detailed Design**

**Objective:** Translate high-level designs into technical blueprints that developers can directly use.

**Activities:**

* **Data Modeling:** We create the **ER diagram** and map it into Salesforce custom objects. For example, the Event\_\_c object includes fields like Event Name, Date, Venue, Capacity, while the Registration\_\_c object links Participants to Events.
* **Relationships:** Registration\_\_c acts as a junction between Event and Participant, ensuring one participant can register for multiple events.
* **UI/UX Mockups:** Low-fidelity wireframes are created for event dashboards, registration forms, and organizer consoles.
* **Security Model:** Define profiles, roles, and permission sets. For instance, Organizers can create and manage events but cannot modify system settings, while Admins have full access.

**Deliverables:** Detailed Object Model, Field Definitions, Security Matrix, and UI Mockups.

**Phase 5: Implementation**

**Objective:** Build the Smart Event Management application inside Salesforce based on the detailed design.

**Activities:**  
Implementation begins with setting up the Salesforce Developer Org. Custom objects such as Event\_\_c, Registration\_\_c, and Participant\_\_c are created with their respective fields. Relationships are established so that each registration is linked to an event and a participant. Page layouts are configured to make data entry intuitive. For instance, the Event page layout displays related registrations, ensuring organizers can see participation at a glance.

Automation is added using **Flows**. A record-triggered flow is created to automatically calculate available seats when a new registration is added. Another flow ensures that participants receive confirmation emails after successful registration. Validation rules are implemented to maintain data integrity, such as preventing event dates from being set in the past. Where advanced logic is required, **Apex classes and triggers** are written, such as ensuring no participant registers twice for the same event.

**Deliverables:** Configured Salesforce Org with working objects, relationships, automation flows, and Apex logic aligned to requirements.

**Phase 6: Testing**

**Objective:** Verify that the application meets requirements, works reliably, and is free of defects.

**Activities:**  
Testing begins with **Unit Testing** of Apex classes, where each piece of logic (like seat availability calculation) is validated using test classes. Salesforce requires at least 75% code coverage before deployment, so all triggers and Apex methods are tested. Next, **System Testing** checks if workflows, flows, and validations behave as expected. For example, trying to register for a past event should throw an error.

**User Acceptance Testing (UAT)** is conducted with mock organizers and participants. Test cases include: creating an event, registering multiple participants, checking notifications, and generating event summary reports. Any defects or mismatches with requirements are logged and fixed. Performance testing ensures the system can handle high traffic during popular event registrations.

**Deliverables:** Test Plan, Test Case Document, UAT Sign-off Report, and Debug Logs.

**Phase 7: Deployment**

**Objective:** Move the application from development/testing to a production-like environment.

**Activities:**  
Since this is a certification project, deployment may stay within a single Developer Org, but the process is documented to simulate a real-world scenario. **Change Sets** or **Salesforce CLI (SFDX)** are used to package the developed components. The deployment checklist ensures that all objects, fields, flows, Apex classes, and permissions are migrated successfully.

Before deployment, data migration is planned. Sample data for events, participants, and registrations is loaded using **Data Import Wizard** or **Data Loader**. Deployment also includes configuring dashboards and reports so that once the system goes live, organizers immediately get insights into participation and revenue.

**Deliverables:** Deployment Plan, Migrated Salesforce Org with sample data, and Deployment Verification Report.

**Phase 8: Maintenance**

**Objective:** Ensure the system continues to run smoothly, adapt to new requirements, and fix issues quickly.

**Activities:**  
Post-deployment, monitoring mechanisms are set up using Salesforce’s **System Audit Trail** and error logs. If issues such as failed flows or unexpected Apex errors occur, they are documented and resolved. Regular backups of event and registration data are scheduled using Salesforce Data Export.

The system is also enhanced based on user feedback. For example, if organizers request the ability to export registrations to Excel, a new feature is added using Salesforce reports. Maintenance also involves reviewing security (role hierarchy, profile permissions) and updating them as organizational needs evolve.

**Deliverables:** Maintenance Logs, Backup Schedules, and Enhancement Request Register.

**Phase 9: Documentation**

**Objective:** Create comprehensive documentation for future users, developers, and administrators.

**Activities:**  
User documentation includes a **User Guide** with step-by-step instructions on creating events, managing registrations, and viewing reports. Screenshots of Salesforce UI are added for clarity. Administrator documentation explains how to manage user roles, adjust automation, and troubleshoot errors. Technical documentation covers data models, object relationships, Apex class descriptions, and flow diagrams.

This documentation ensures continuity: if a new developer or admin takes over the system, they can understand how it works without starting from scratch. It also supports audit and compliance, showing that processes are transparent and properly managed.

**Deliverables:** User Guide, Admin Guide, Technical Design Document, and System Architecture Document.

**Phase 10: Final Presentation & Handover**

**Objective:** Showcase the project outcomes, demonstrate features, and officially hand over the system.

**Activities:**  
The project is summarized in a **final presentation** that follows the structure: Problem Statement → Requirements → Design → Implementation → Demo → Benefits. During the demo, a live walkthrough is given: creating an event, registering a participant, auto-triggering confirmation, and showing a dashboard report.

The presentation also highlights how Salesforce was leveraged (Flows, Apex, Reports, Dashboards) and why this solution is effective for event management. At the end, the system is handed over to the mentor/evaluator with all documentation, credentials (if needed), and instructions for usage. Feedback is collected and incorporated into the final submission.

**Deliverables:** Final Project Presentation, Demo Video (if required), and Handover Package with all documents.