TABLE OF CONTENTS

Chapter 0: Synopsis

Title of the project

Volunteer Management

Statement about the problem

Organizations and NGOs often struggle with efficiently managing volunteer engagement for their social welfare programs. Without a centralized platform, it's challenging for them to effectively communicate upcoming events and opportunities to potential volunteers. Similarly, volunteers face difficulties in discovering relevant events and coordinating their participation due to the lack of a centralized resource. This fragmented approach leads to missed opportunities, decreased volunteer engagement, and inefficient utilization of resources for social welfare initiatives.

Why this topic?

Creating a Volunteer Management System for social welfare programs addresses a critical need in our community. By providing a centralized platform for organizations, NGOs, and volunteers, we aim to bridge the gap between those in need of support and those willing to contribute their time and skills. This project aligns with the values of community engagement, social responsibility, and effective resource utilization, making it a compelling and impactful endeavor. Through this system, we can streamline volunteer engagement, enhance the effectiveness of social welfare programs, and ultimately contribute to positive social change.

Objective and Scope

Scope:

The scope of the system is to develop a robust volunteer management system that facilitates seamless communication and co-ordination between organisations , NGO's and volunteers. The scope of the system is limited to Mumbai region, catering to organisations of various type including Non-Profit, Commercial, Educational, Government organisations, Healthcare, Charitable, Religious, Social organisations. This ensures that the system is tailored to specific needs and context of Mumbai community while being inclusive and accessible to a diverse range of organisations seeking volunteer support for their social welfare programs.

Objectives:

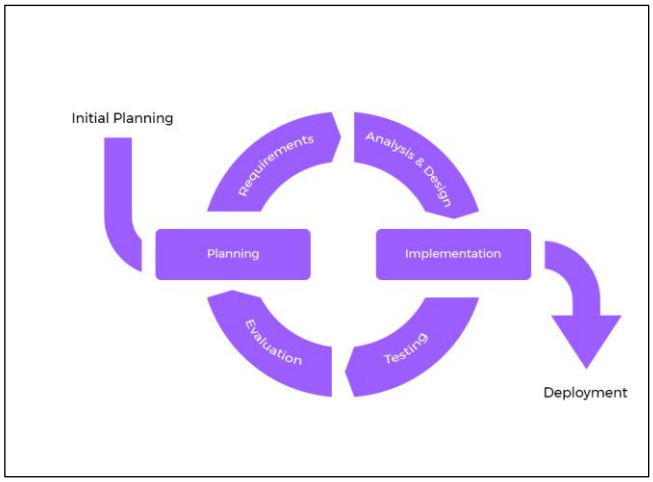
i. Provide a user-friendly interface for organisations and NGO's to post events , manage Volunteer requirements , and track volunteer engagement.
ii. Enable volunteers to easily discover , apply for, and participate in relevant social welfare events based on their interests, skills and availability.
iii. Improve efficiency in volunteer management , event co-ordination and resource allocation for social welfare initiatives.
iv. Foster a sense of community , collaboration and social impact among volunteers and organisations

v. Improving social interactions between volunteer by allowing them to participate and communicate with a diverse variety of people.

Methodology

For this System, Iterative development model would be more suitable because of the following reasons:

- Will allow continuous refinement and improvement based on feedback and evolving requirement.
- Offers flexibility in accommodating new features , addressing issues and incorporating user feedback at various stages.
- Encourages active involvement of stakeholders

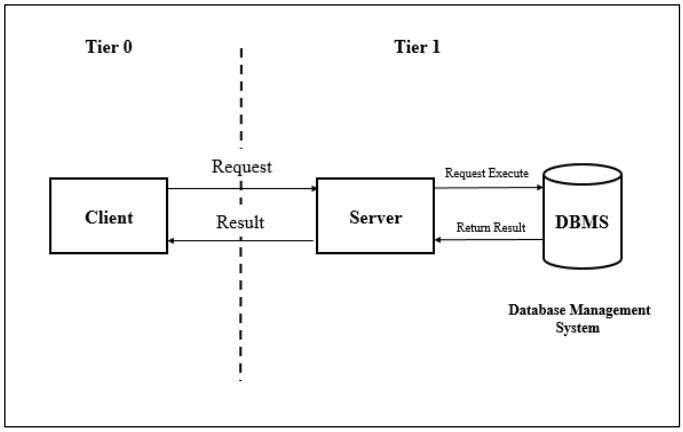


Proposed Architecture

The Project would be developed using a Client-Server 2 Tier Architecture, means organizing website in 2 main parts:

i. Client:
The User Interface and Presentation of the system will run on user's device. Client is responsible for initiating request to server.

ii. Server:
The Server component hosts the Application Processing and Data Storage, It responds to requests by processing the requests.



Requirements

1. Software Requirements:
    a. Database: MySQL
    b. Back-End: PHP
    c. Front-End: HTML,CSS, Javascript (Jquery), Bootstrap, AJAX

2. Hardware Requirements:
    a. A Basic computer with an decent processor such as intel i3 10^{th} Gen or 11^{th} Gen or AMD Ryzon 3 Processor
    b. Ram: Minimum 4GB
    c. HardDisk: 128GB or 256GB SSD, 4GB of available space
    d. Monitor: 1200x800 Minimum screen resolution

3. Platform:
    a. Visual Studio Code (VS Code)

Contribution To Society:

The Volunteer Management System contributes significantly to society in several ways:

1. Facilitating volunteer engagement and collaboration among organizations, NGOs, and volunteers.

2. Optimizing resource allocation for social welfare programs, enhancing their impact.

3. Improving communication, coordination, and feedback mechanisms for better program outcomes.

4. Fostering a culture of collaboration, knowledge sharing, and collective action within communities.

5. Empowering individuals to contribute meaningfully to social causes, promoting civic engagement and social responsibility.

Conclusion:

In conclusion, the development of a Volunteer Management System represents a significant step towards fostering a more connected, efficient, and impactful approach to social welfare initiatives. By bringing together volunteers, organizations, and NGOs on a centralized platform, the system streamlines volunteer engagement, optimizes resource utilization, and enhances the overall effectiveness of social welfare programs.

Through improved communication, collaboration, and empowerment of individuals, the system contributes to positive social change, stronger communities, and a more inclusive society.

Chapter 1 :Introduction

1.1 Background

Traditionally, organizations have relied on methods such as word-of-mouth, flyers, local media advertisements, volunteer fairs, and referrals to recruit volunteers for their social welfare programs. While these methods are still relevant, they face several challenges in effectively managing volunteer engagement. These challenges include limited reach, resource-intensive processes, lack of centralization in information, communication barriers, and difficulties in tracking and reporting volunteer activities.

In response to these challenges, the proposed Volunteer Management System website aims to revolutionize volunteer hiring and management practices. By centralizing information about volunteer opportunities, providing efficient communication tools, automating processes such as volunteer matching and application management, implementing robust data tracking and reporting features, and expanding reach to a wider audience of potential volunteers, the website addresses the shortcomings of traditional methods. It streamlines processes, improves communication and coordination, enhances the volunteer experience, and ultimately contributes to more effective and impactful social welfare initiatives in the Mumbai region.

1.2 Objectives

Provide a centralized platform for organizations and NGOs to post volunteer opportunities for social welfare programs in the Mumbai region.

i. Provide a centralized platform for organizations and NGOs to post volunteer opportunities for social welfare programs in the Mumbai region.
ii. Enable volunteers to easily find and apply for relevant volunteer positions based on their skills, interests, and availability.
iii. Implement efficient communication tools to facilitate seamless interaction between organizations and volunteers, improving coordination and reducing communication barriers.
iv. Expand the reach of volunteer opportunities to a wider audience, including non-profit, commercial, educational, and other organizations, fostering collaboration and collective impact for positive social change.
v. To reduce the time and efforts to search and recruit potential volunteer for a social welfare program by organizations.

1.3 Scope, Purpose and Applicability

1.1 Purpose:

The purpose of the Volunteer Management System website is to streamline volunteer engagement, enhance communication between organizations and volunteers, and Encourage community involvement and social responsibility by providing individuals and organizations with opportunities to contribute meaningfully to social causes and make a positive difference in their communities.

1.2 Scope:

The scope of the system is to develop a robust volunteer management system that facilitates seamless communication and co-ordination between organizations of various types such as non-profit, commercial, educational, government, religious organisations and volunteer's for social welfare programs across Mumbai region.

1.3 Applicability:

The applicability of the Volunteer Management System website extends to various types of organizations and entities in the Mumbai region, including:

1. **Non-Profit Organizations:** Non-profit organizations focused on social causes such as education, healthcare, environmental conservation, and community development can use the platform to recruit volunteers for their programs and initiatives.
2. **Commercial Entities:** Commercial entities with corporate social responsibility (CSR) initiatives can leverage the platform to engage employees in volunteering activities, support local communities, and contribute to social welfare programs.
3. **Educational Institutions:** Schools, colleges, and universities can utilize the platform to involve students, faculty, and staff in community service projects, awareness campaigns, and educational outreach programs.
4. **Community Groups:** Local community groups, neighbourhood associations, and grassroots organizations can use the platform to organize volunteer-led activities, events, and campaigns for community improvement and social change.
5. **NGOs and Charitable Foundations:** NGOs, charitable foundations, and advocacy groups working on various social issues such as poverty alleviation, human rights, and disaster relief can utilize the platform to recruit volunteers, coordinate relief efforts, and raise awareness.

The Volunteer Management System website is designed to be adaptable and accessible to a wide range of organizations and entities, offering a centralized platform for efficient volunteer management,

collaboration, and collective impact in addressing societal challenges and promoting positive social outcomes in the Mumbai region.

Chapter 2 : Survey of Technology

Front End Languages:

- **HTML5 :**
  HTML5 is a markup language used for structuring and presenting content on the World Wide Web. HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves, and rationalizes the markup available for documents and introduces markup and application programming interfaces (APIs) for complex web applications.

- **JavaScript:**
  JavaScript is a versatile programming language used for creating interactive and dynamic web content. It runs on the client side, enabling functionalities like form validation, DOM manipulation, and AJAX requests. JavaScript is essential for web development as it adds interactivity to websites, creates responsive user interfaces, handles events, and communicates with servers asynchronously.

- **JQuery:**
  jQuery is a fast, lightweight, and feature-rich JavaScript library that simplifies DOM manipulation, event handling, and AJAX interactions. It provides a concise and efficient way to write JavaScript code by streamlining common tasks like selecting elements, animating content, making AJAX requests, and handling events across different browsers. jQuery abstracts complex JavaScript functionalities into simple methods, improving code readability and reducing development time.

- **CSS (Cascading Style Sheets):**
  CSS is a style sheet language used for styling and formatting web documents written in HTML or XML. It controls the visual presentation of web pages, including layout, colors, fonts, and spacing. CSS separates content from design, allowing developers to create visually appealing and responsive layouts by applying styles to HTML elements, classes, IDs, and pseudo-classes.

- **Tailwind CSS:**
  Tailwind CSS is a utility-first CSS framework that provides a set of pre-designed utility classes to style HTML elements. It focuses on rapid development and customization without writing custom CSS. Tailwind CSS allows developers to apply styles directly in HTML using classes like `bg-blue-500`, `text-lg`, `p-4`, defining background colors, text sizes, padding, margins, and more. It promotes a scalable and maintainable approach to styling web applications by composing utilities to create custom designs.

- **ASP.NET:**
  ASP.NET is a web application framework developed by Microsoft for building dynamic and scalable web applications, APIs, and services using languages like C# and VB.NET. It provides a robust set of tools, libraries, and APIs for web development, including MVC (Model-View-Controller) and Web Forms for building web applications, Web API for creating RESTful services, and ASP.NET Core for cross-platform development. ASP.NET supports features like authentication, authorization, routing, caching, and database access, making it a comprehensive platform for web development.

- **React**
  React is a JavaScript library for building user interfaces, primarily used for developing the front end of web applications. It allows developers to create reusable UI components and manage the state of the application efficiently. React follows a component-based architecture and uses a virtual DOM for optimal performance, enabling fast rendering and updates. It is maintained by Facebook and has a large and active community, along with extensive documentation and ecosystem of tools and libraries.

- **Angular:**
  Angular is a TypeScript-based web application framework maintained by Google, used for building dynamic and interactive single-page applications (SPAs). It follows the MVC (Model-View-Controller) architecture and provides features like data binding, dependency injection, routing, and form handling out of the box. Angular offers a comprehensive suite of tools and libraries for testing, debugging, and

optimizing applications, making it suitable for large-scale enterprise applications with complex requirements.

Back End Languages:

- **Next.js**
  Next.js is a React framework that provides functionality such as server-side rendering, routing, and generating static websites for React-based web applications. It simplifies the development process by offering built-in features like automatic code splitting, hot module replacement, and server-side rendering, enhancing performance and SEO. Next.js is highly flexible and scalable, allowing developers to build complex applications with ease while providing a great developer experience.

- **PHP:**
  PHP is a server-side scripting language designed for web development. It is widely used for creating dynamic web pages and web applications. PHP code is embedded within HTML, allowing developers to mix logic with presentation. It supports various databases and frameworks like Laravel, CodeIgniter, and Symfony, making it versatile for building robust backend systems.

- **Ruby:**
  Ruby is a dynamic, object-oriented programming language known for its simplicity and productivity. It is commonly used with the Ruby on Rails framework for web development. Ruby on Rails provides conventions over configurations, making it efficient for building database-backed web applications. Ruby's readability and expressiveness contribute to its popularity among developers.

- **Node.js**
  Node.js is a runtime environment that allows developers to run JavaScript code server-side. It uses an event-driven, non-blocking I/O model, making it lightweight and efficient for handling concurrent requests. Node.js is commonly used for building scalable, real-time web applications and APIs. It has a large ecosystem of packages available through npm (Node Package Manager).

- **Rust:**

Rust is a systems programming language known for its focus on safety, performance, and concurrency. While not as commonly used for traditional web development as other languages, Rust can be used for backend development, especially for performance-critical applications or services. Its strong type system and memory safety features make it suitable for building robust and secure systems.

- **Flask:**
  Flask is a lightweight Python web framework designed for simplicity and flexibility. It provides tools and libraries for building web applications, APIs, and microservices. Flask follows a minimalist approach, allowing developers to choose components and extensions as needed. It is often used for prototyping, small-scale projects, and rapid development of backend systems.

Database:

- **MySQL:**
  MySQL is a relational database management system (RDBMS) developed by Oracle that is based on structured query language (SQL). MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL provides an implementation of a SQL database very well suited for small to medium web pages. A database is just a structured collection of data that is organized for easy use and retrieval. Common applications for MySQL include php and java-based web applications that require a DB storage backend.

- **PostgreSQL:**
  PostgreSQL is a powerful, open-source object-relational database system known for its reliability, robustness, and extensibility. It uses SQL (Structured Query Language) for querying and managing data, offering features like transactions, foreign keys, triggers, and stored procedures. PostgreSQL supports various data types, including numeric, text, JSON, and arrays, making it versatile for different types of applications. It's widely used in enterprise-level applications due to its scalability, support for complex queries, and ACID compliance, ensuring data integrity.

- **MongoDB:**
  MongoDB is a popular NoSQL database that stores data in flexible, JSON-like documents, making it easy to store and manage unstructured

data. It offers high scalability, performance, and flexibility, making it suitable for handling large volumes of data in real-time applications. MongoDB's document model allows for quick and easy data retrieval, and it supports features like indexing, shading, and replication for scalability and fault tolerance. It's commonly used in modern web and mobile applications, IoT (Internet of Things), and analytics applications requiring flexible data storage and retrieval capabilities.

- **AWS Dynamo DB:**
  As part of a larger ecosystem of Amazon Web Services (AWS), this database is a NoSQL database that is known for its speed and efficiency when it comes to retrieval of information and data from the system. It is a key-value database, so there are variety of data types that can be stored within this system. It is a highly scalable and a complex database system for developers that are working with the applications that need to manage big user data and constant engagement.

Why these language?

The system will be developing using Html5, Tailwind CSS and Jquery for the user-friendly front-end, Php (Laravel) for the back-end, and MySQL for the database.

HTML , jQuery And Tailwind CSS

- **HTML:** provides the foundational structure of web pages, essential for any web application.
- **jQuery:** is essential for creating interactive and dynamic user interfaces. It simplifies JavaScript programming, making it easier to handle events, create animations, and manage HTML document traversing and manipulation.
- **Tailwind CSS:** is crucial for styling web pages, ensuring that the website is visually appealing and user-friendly. it allows for rapid development with its utility-first approach, providing a flexible and customizable design system that can be adapted to various project requirements.

Why PHP ?

It is a server-side scripting language that is particularly well-suited for web development. It is widely used due to its ability to interact with databases, handle sessions, and manage dynamic content. PHP is also compatible with

most web servers and databases, making it a versatile choice for backend development.

Why MySQL ?

It is chosen for its reliability, performance, and ease of use. It supports large-scale applications and can handle high volumes of data and transactions. MySQL's robust features, such as data security, backup, and recovery, make it a dependable choice for managing the database of the Volunteer Management System.

Chapter 3:Requirements and Its Analysis

3.1 Problem Definition:

Organizations, NGOs, and volunteers often face challenges in efficiently connecting and collaborating for social welfare programs. Traditional methods of volunteer recruitment and management are often fragmented, resource-intensive, and lack effective communication channels. This leads to difficulties in matching volunteers with suitable opportunities, coordinating activities, tracking engagement, and maximizing the impact of social welfare initiatives. As a result, there is a need for a centralized and user-friendly Volunteer Management System website that streamlines volunteer engagement, enhances communication and coordination, optimizes resource allocation, and promotes community involvement for positive social impact.

3.1.1 Sub-Systems

1. User Management :

-Allow users (organizations and volunteers) to register on the platform and manage their profiles with information such as skills, interests, availability, and contact details.

- Implement secure authentication mechanisms to verify user identity and control access to different system features based on user roles (admin, organization, volunteer).

2. Organization management:

- Allows organization to register and create profile.
- Providing functionalities for posting volunteer opportunities.
- Enable management of event applications.
- Offer communication tools to connect with volunteers.

3. Volunteer Management:

- Allows Volunteers to register and create profiles.
- Provides opportunities to search and browse volunteer events based on preferences (location, cause, timing, skills).
- Enables applying for volunteering positions.
- Tracks volunteer hours and engagement.

4. Communication Sub-System:
- Provide communication tools such as messaging, email notifications, and alerts to facilitate seamless interaction between organizations and volunteers, keeping them informed about opportunities, updates, and reminders.

5. Feedback and Evaluation Sub-System:
- Collect feedback from volunteers and organizations about their experiences, satisfaction levels, and suggestions for improvement.
- Evaluate the impact of volunteer activities and social welfare programs, gathering insights to measure success and identify areas for enhancement.

6. Admin and Management :
- Provide an admin dashboard with functionalities for user management, content moderation, data administration, and system configuration.
- Implement moderation tools to review and approve volunteer opportunities, user-generated content, and communications for compliance and quality assurance.

These sub-systems work together to create a comprehensive Volunteer Management System website that enhances volunteer engagement, communication, coordination, tracking, and impact assessment for social welfare programs in the Mumbai region.

3.2 Requirement Specification:

3.2.1 Requirement Gathering:

- **Method Used:**
  1. **Survey / Questionnaire:**
     Questionnaires and surveys are basically a set of questions that are distributed among the stakeholders in order to get a understanding of the system. The questionnaires can be distributed to multiple stakeholders at the same time, hence saving time and reducing the efforts required to gather information.

  2. **Interview:**

Interviews are usually conducted one -on -one. Interview provide the flexibility of asking follow-up questions which can help gather more specific and detailed information.

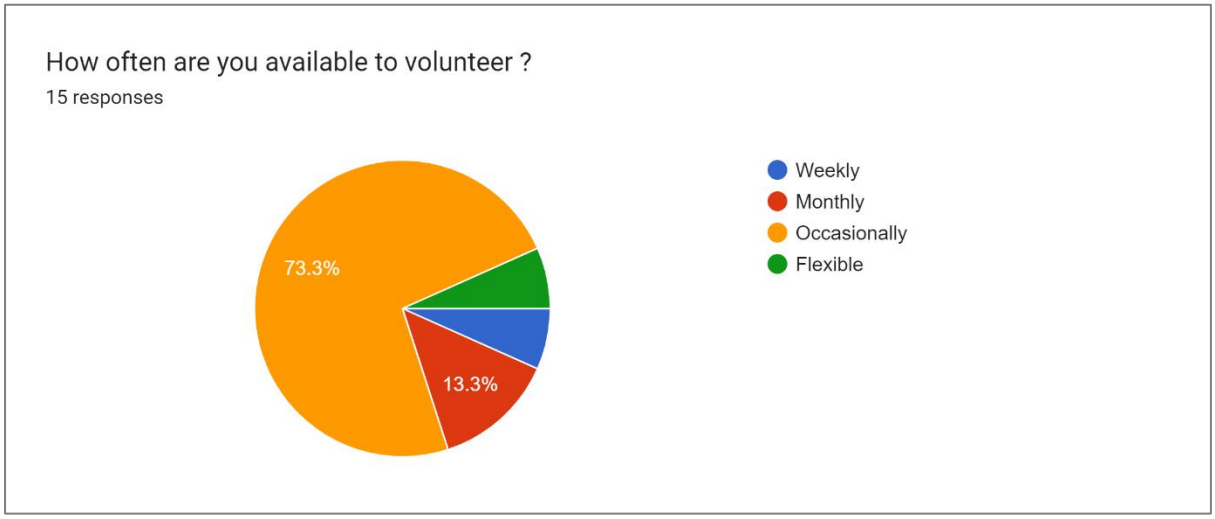
- **The Questions asked in the survey are as follows:**
  1. How often are you available to volunteer ?
  2. What types of volunteer opportunities are you interested in ?
  3. How do you prefer to receive information about volunteer opportunities ?
  4. Would you prefer to use a volunteer management website that focuses specifically on volunteering opportunities in Mumbai region ?
  5. What challenges or difficulties do you typically face in finding and applying for volunteering positions ?
  6. What features or functionalities would make it easier for you to discover, apply for, and participate in volunteer opportunities ?
  7. Would you be interested in using a website that suggest volunteering opportunities based on your skills and interests ?
  8. How important is it for you to receive feedback and recognition for your volunteering contributions ?
  9. Would you be interested in networking or collaborating with other volunteers or organizations through the volunteer management website ?
  10. Do you have any suggestions for additional features that would improve your volunteer experience ?

- **The Questions asked in the Interview are as follows:**
  1. What types of social welfare programs do you run ?
  2. On average, how many volunteers do you manage per programs ?
  3. What are the key challenges or difficulty you face in volunteer management and co-ordination ?
  4. What are the features you expect from a volunteer management website ?
  5. How do you currently recruit and manage volunteers for your programs ?
  6. What information do you typically include when creating a volunteer opportunity ?

7. What information do you need from volunteers to match them with suitable opportunities ?
8. What are your preferred communication methods with volunteers ?
9. Are there any legal or compliance requirements related to volunteer management that need to be considered ?
10. What would be your preferred method of collecting volunteer applications ?
11. Do you require any screening (background checks) for specific programs ?
12. Do you have any suggestions that you would like to see in the system to enhance volunteer management and program effectiveness ?

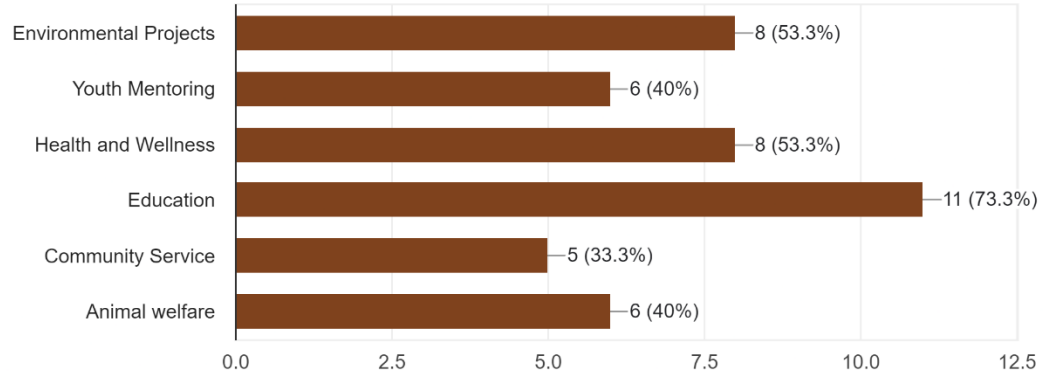
- **Link of google forms for Volunteers :**
  https://forms.gle/YAMyNjGK1pxDUyrE7

  Followings are the results:



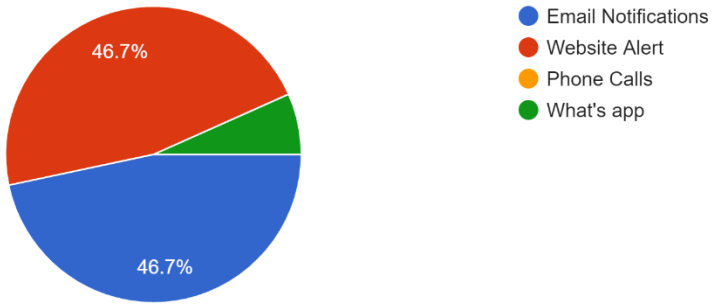
What types of volunteer opportunities are you interested in ?

15 responses



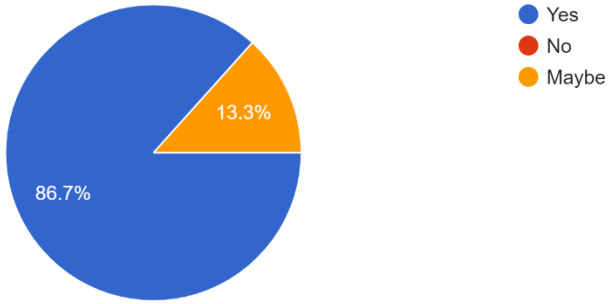
How do you prefer to receive information about volunteer opportunities ?

15 responses



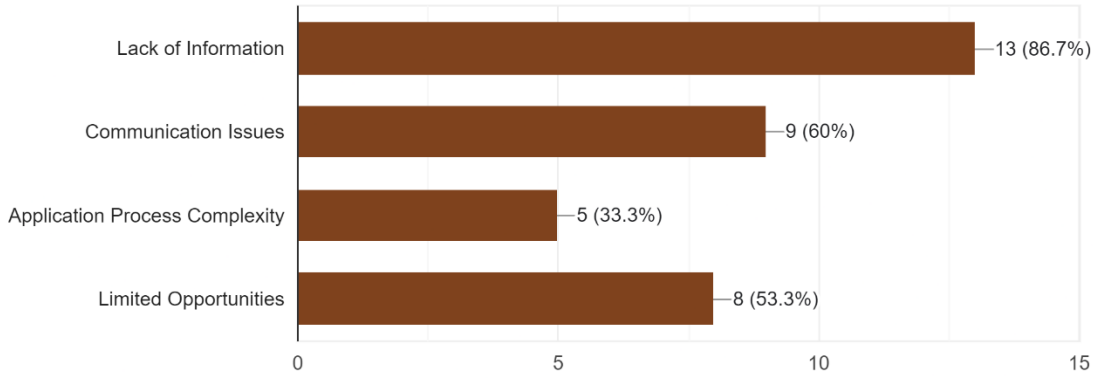
Would you prefer to use a volunteer management website that focuses specifically on Volunteering Opportunity in Mumbai Region ?

15 responses



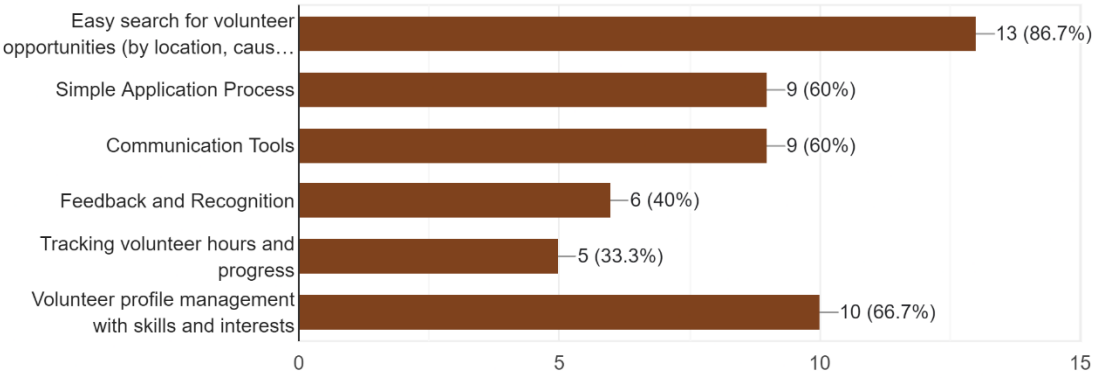
What challenges or difficulties do you typically face in finding and applying for volunteering positions?

15 responses



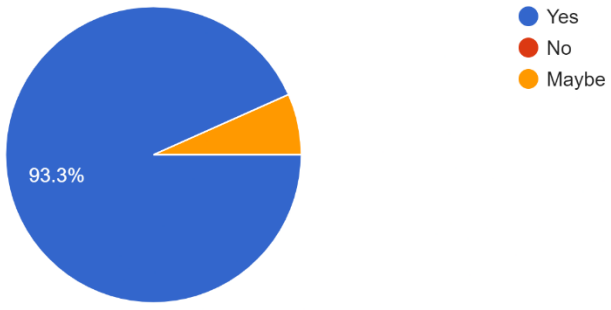
What features or functionalities would make it easier for you to discover, apply for, and participate in volunteer opportunities?

15 responses



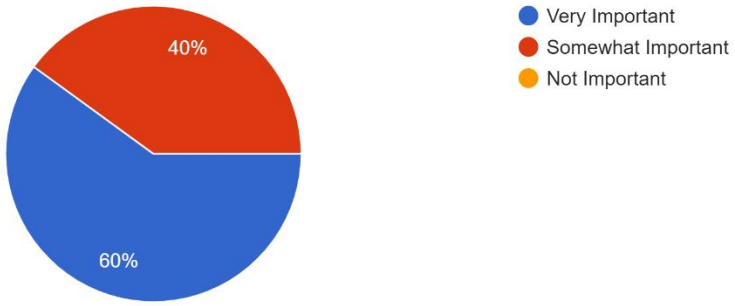
Would you be interested in using a website that suggests volunteering opportunities based on your skills and interest ?

15 responses



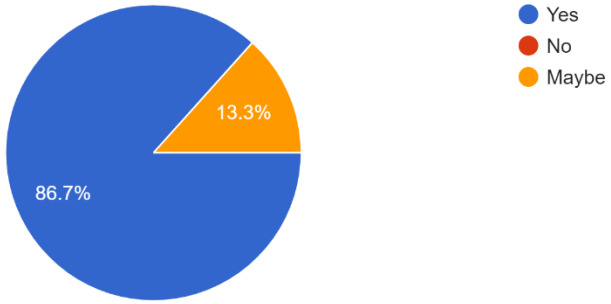
How important is it for you to receive feedback and recognition for your volunteer contributions?

15 responses



Would you be interested in networking or collaborating with other volunteers or organizations through the Volunteer Management Website?

15 responses



Do you have any suggestions for additional features that would improve your volunteer experience?

15 responses

None

no

I would like to have a seearching functionality where i can easily find a volunteer opportunity according to my ease and should me able to chat with them and contact them

Not now

How about a feature that lets volunteers easily track their hours and tasks completed, with automatic reminders for upcoming shifts? You could also add a section for volunteers to share feedback or ideas for improvement with the organization.

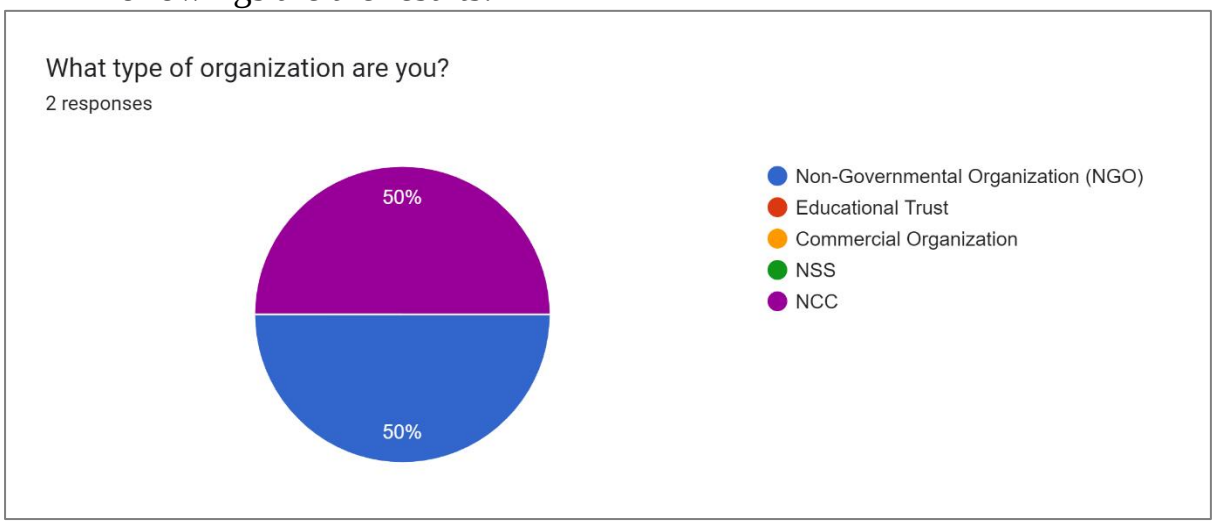
Good user interface and valid or authenticate information about volunteering job

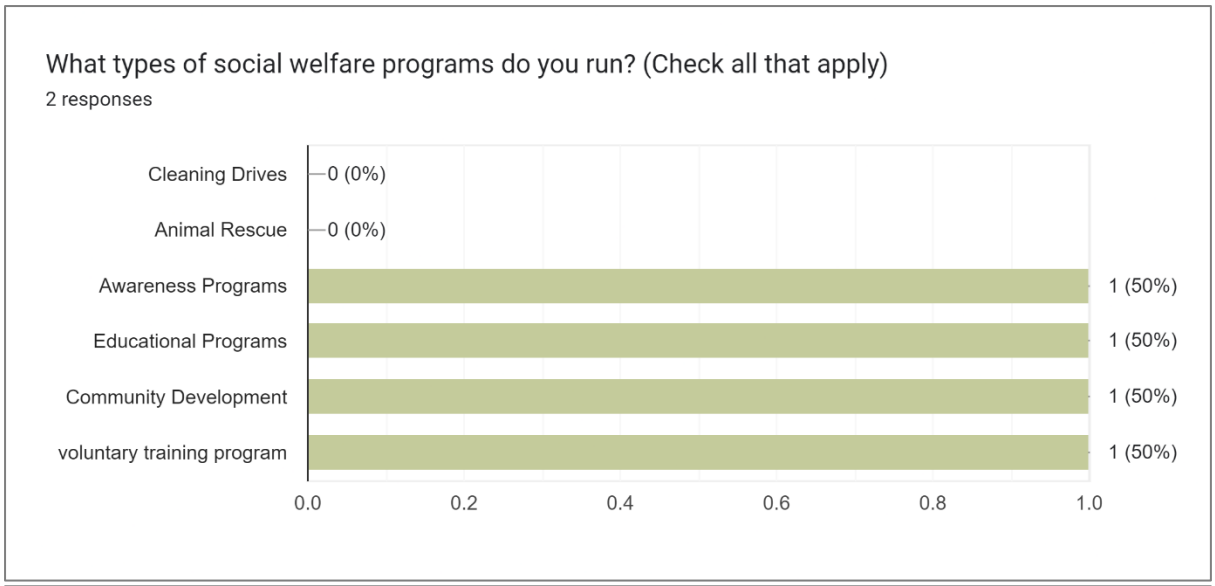
No go ahead for development

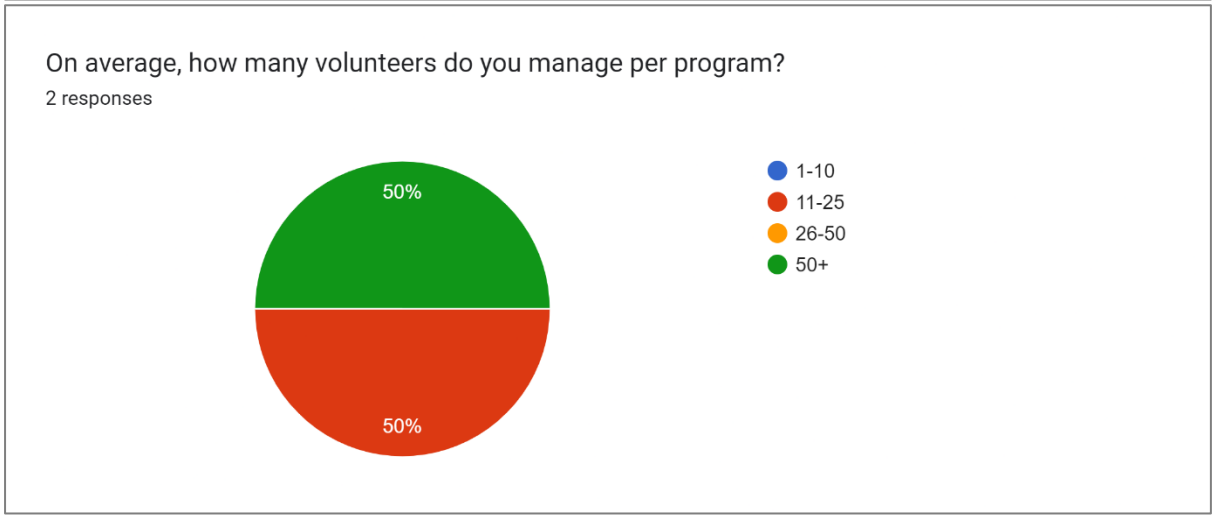
-

- **Link of google forms for Organization:** https://forms.gle/5cQ2pMD61sqP88Je9

Followings are the results:

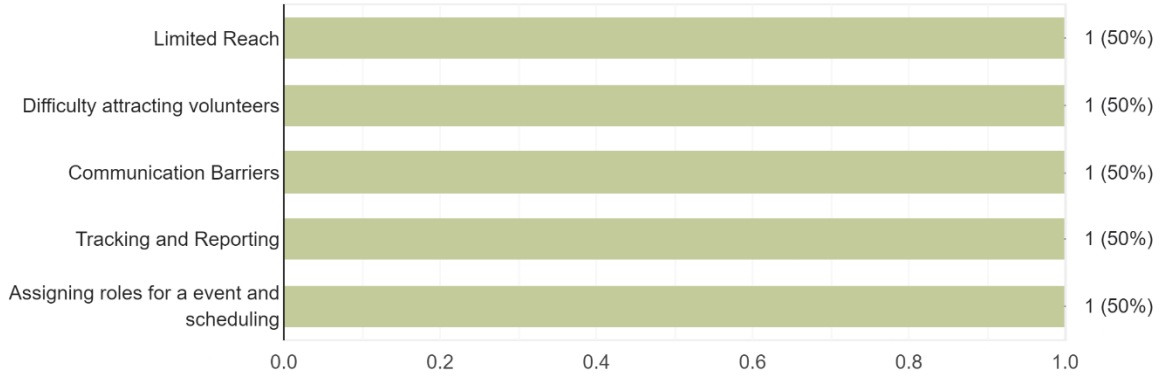






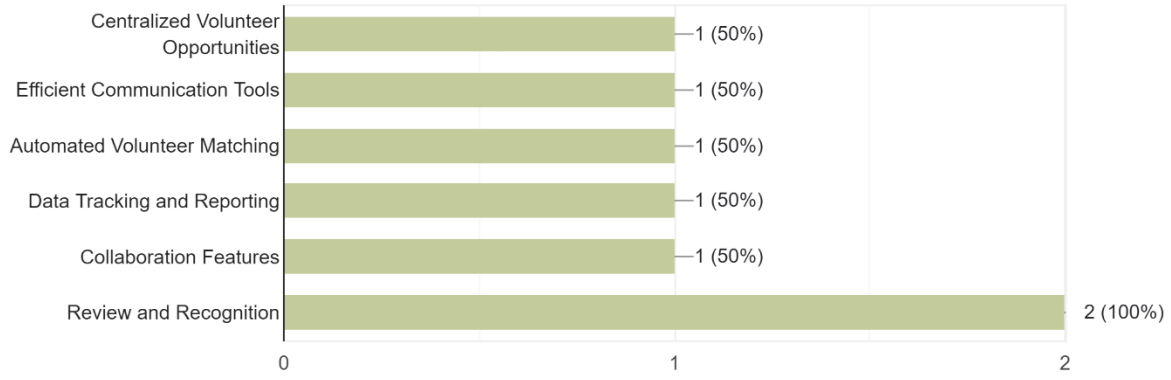
What are the key challenges or difficulty you face in volunteer management and coordination?

2 responses



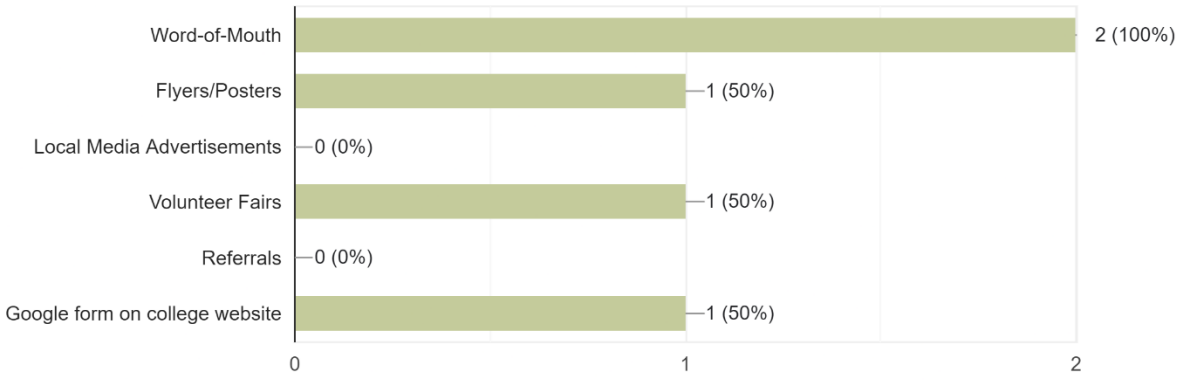
What are the features you Expect from a Volunteer Management Website ?

2 responses



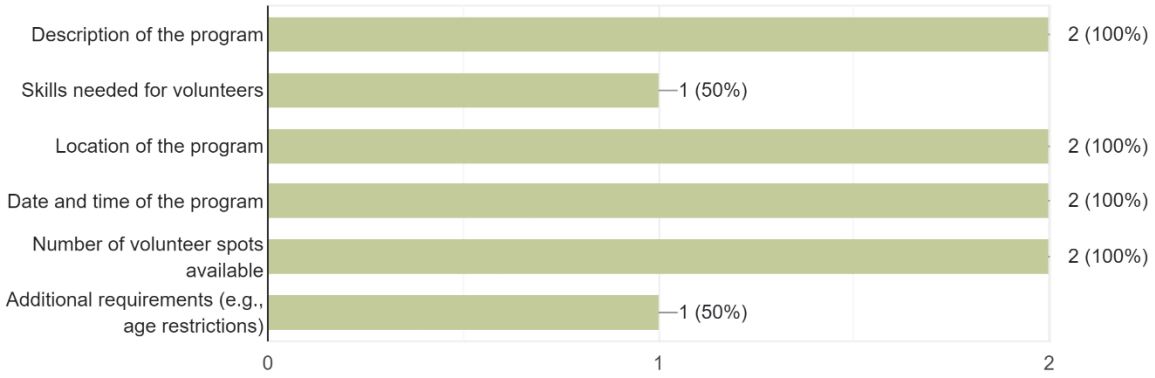
How do you currently attract and manage volunteers for your programs?

2 responses



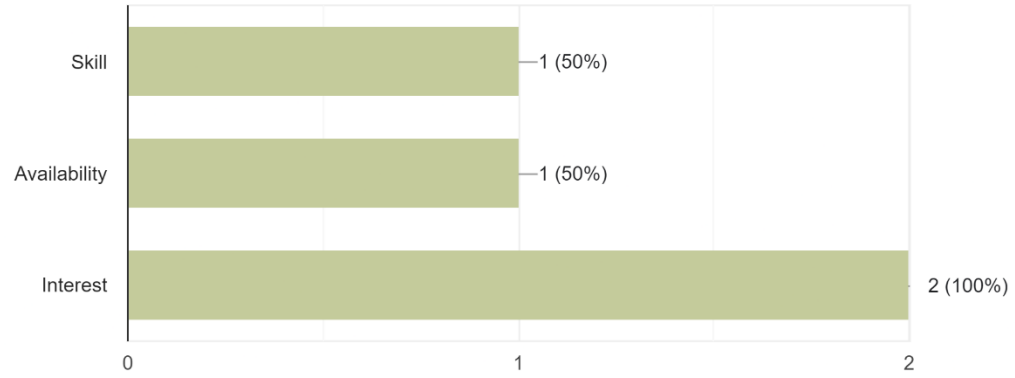
What information do you typically include when creating a volunteer opportunity? (Check all that apply)

2 responses



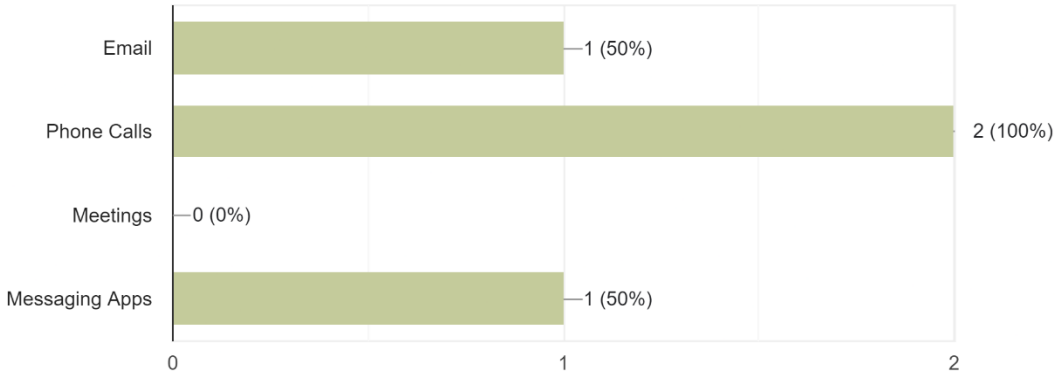
What information do you need from volunteers to match them with suitable opportunities?

2 responses



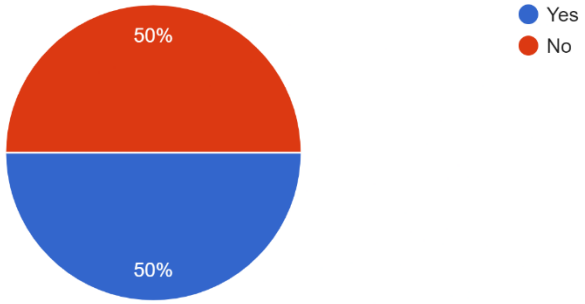
What are Your Preferred Communication Methods with Volunteers ?

2 responses



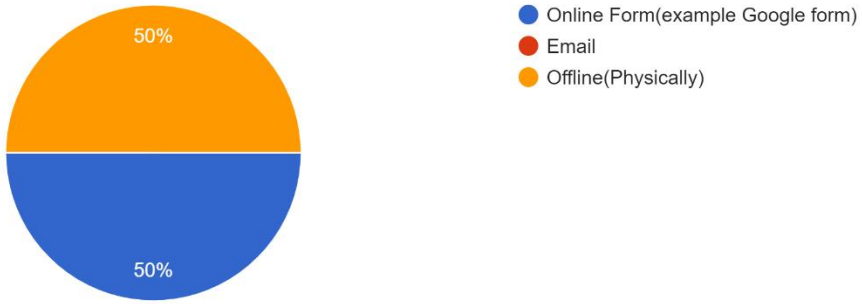
Are there any legal or compliance requirements related to volunteer management that need to be considered?

2 responses



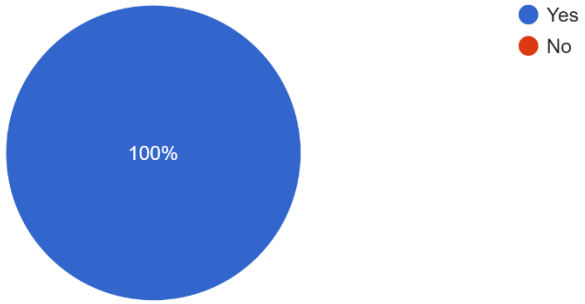
What would be your preferred method of collecting volunteer applications? (Choose one)

2 responses



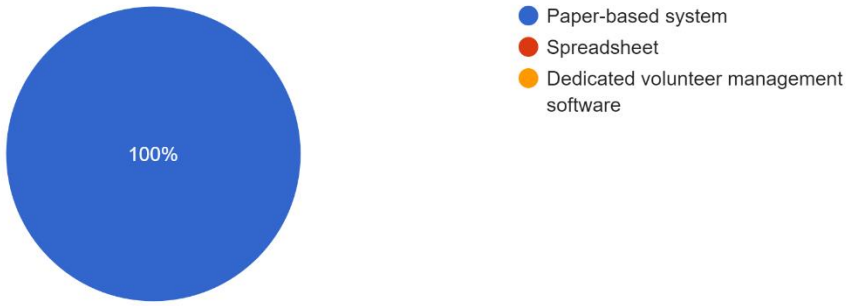
Do you require any screening (background checks, references) for specific programs? (Choose one)

2 responses



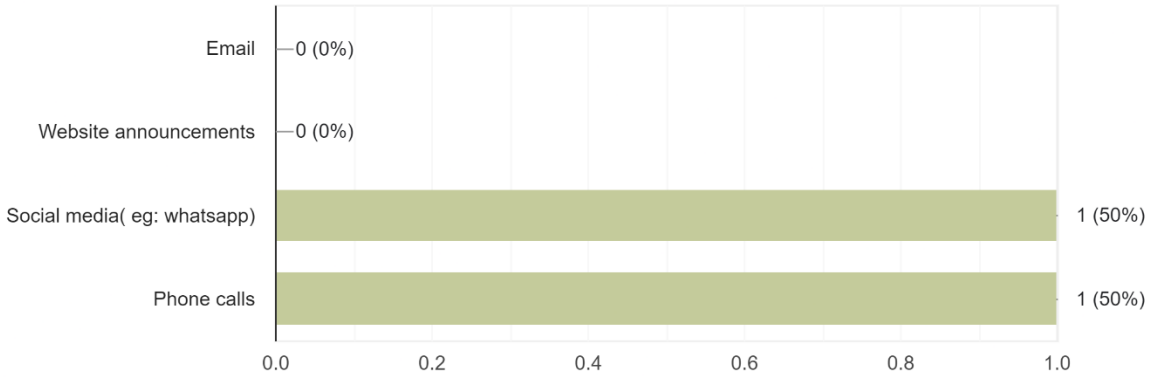
How do you currently track volunteer hours and engagement? (Choose one)

2 responses



How do you currently communicate with volunteers about program updates and changes? (Check all that apply)

2 responses



Do you have any suggestions that you would like to see in the system to enhance volunteer management and program effectiveness?

2 responses

An image gallery for each organisation to showcase their past programs

To notify them about there roles , where they are posted and what time they have to arrive.

3.2.2 Requirement Analysis

By analysing the Google Form responses, we gathered insightful data about the needs and preferences of both organizations and volunteers for the Volunteer Management System.

Organization Feedback:

Organizations mostly have difficulty in finding and recruiting volunteers for their events. Almost 90% of organizations prefer to switch to an online system for managing volunteer registrations.
50% of organizations currently do not have a dedicated platform for volunteer management.

Challenges faced by organizations include:

- Difficulty in reaching a wider audience.
- Inefficiencies in managing volunteer data manually.
- Lack of tools for effective communication with volunteers.

Additional desired features include:

- 75% want a communication platform to engage with volunteers.
- 70% want tools for feedback collection and reporting.

Volunteer Feedback:

- 75% of volunteers have experienced difficulty in finding suitable volunteer opportunities.
- 92% of volunteers prefer an online platform to browse and apply for volunteer opportunities.

Challenges faced by volunteers include:

- Lack of information about available opportunities.
- Difficulty in applying for events through traditional methods.
- Limited communication with organizations.

85% of volunteers want an easy-to-use platform for browsing and applying to events.
80% of volunteers want notifications for new volunteer opportunities.
Most of the volunteers desire a feature for receiving feedback and recognition from organizations.

Additional desired features include:

- 65% want to see reviews and ratings of organizations from past volunteers.

- 60% prefer a Communication Feature to Socialize with others volunteers

3.2.2.1 Functional Requirements

1. User Registration and Login:
   Organizations and volunteers must be able to register and create profiles. Secure login system for all users.

2. Volunteer Matching:
   Automated matching of volunteers to opportunities based on skills and interests.

3. Volunteering Event Management:
   - Organizations can create, manage, and edit event listings.
   - Volunteers can browse and apply for events.

4. Communication Tools:
   In-built messaging system for communication between organizations and volunteers.

5. Notifications and Alerts:
   Email and push notifications for new volunteer opportunities and updates.

7. Feedback and Reviews:
   - Volunteers can leave reviews and feedback for organizations.
   - Organizations can provide feedback and recognition to volunteers.

8. Mobile Compatibility:
   Mobile-friendly interface for accessing the platform on various devices.

9. Socialization Tool:

   Providing a features for volunteer and organisation to post images and comments related to volunteering activity.

3.2.2.2 Non-Functional Requirements

1. Scalability:
   The system must be able to handle an increasing number of users and data without performance degradation.

2. Security:
   - User data must be protected with encryption and secure authentication mechanisms.
   - Regular security audits to ensure the system's integrity.

3. Usability:
   - The platform should have an intuitive and user-friendly interface.
   - Clear navigation and accessible design for users.

4. Maintainability:
   - The codebase should be well-documented and modular to facilitate maintenance and updates.

6. Compatibility:
   - The system must compatible to all type of devices (Mobile phone, Laptop, Desktop PC etc)

3.2.2.3 System Requirement

1. **Registration:**
   Description: The user must first register an account in order to use the site.
   Input: Name, phone, email, password, confirm password.
   Source: User (Organization and Volunteer).
   Output: Input data is stored in the database.
   Destination: Database.
   Action: After registration, an account for the user will be created.
   Pre-condition: User should not have an existing account.
   Post-condition: User can log in to the account with the registered email and password.

Exception: An error message is shown if any input is empty, email or phone number doesn't match the pattern, or if the account already exists.

2. **Login:**
   Description: The user must log in to access their account.
   Input: Email, password.
   Source: User (Organization and Volunteer).
   Output: User authentication.
   Destination: Database.
   Action: User is granted access to their account.
   Pre-condition: User must have a registered account.
   Post-condition: User is logged in and redirected to their dashboard.
   Exception: An error message is shown if the email or password is incorrect.

3. **Event Creation (For Organizations):**
   Description: Organizations can create new events.
   Input: Event name, description, date, time, location, number of volunteers needed, required skills.
   Source: Organization.
   Output: Event details are stored in the database.
   Destination: Database.
   Action: Event is created and listed on the platform.
   Pre-condition: Organization must be logged in.
   Post-condition: Event is visible to volunteers for application.
   Exception: An error message is shown if any input is empty or does not meet validation criteria.

4. **Event Browsing and Application (For Volunteers):**
   Description: Volunteers can browse and apply for events.
   Input: Event name or Type for browsing; Event ID for application.
   Source: Volunteer.
   Output: List of available events for browsing; application details stored in the database for event application.
   Destination: Database.
   Action: Volunteer can view event details and apply for events.

Pre-condition: Volunteer must be logged in.
Post-condition: Volunteer's application is recorded.
Exception: An error message is shown if the application cannot be processed.

5. **Volunteer Matching:**
   Description: The system automatically matches volunteers to events based on their skills and interests.
   Input: Volunteer profile information, event requirements.
   Source: Database.
   Output: List of matched events for volunteers.
   Destination: Volunteer's dashboard.
   Action: Volunteers are notified of suitable events.
   Pre-condition: Volunteer profile must be complete with skills and interests.
   Post-condition: Matched events are displayed to the volunteer.
   Exception: An error message is shown if no matching events are found.

6. **Communication Tool:**
   Description: Allows communication between organizations and volunteers.
   Input: Message content.
   Source: User (Organization and Volunteer).
   Output: Message is stored and displayed to the recipient.
   Destination: Database.
   Action: Message is sent and received.
   Pre-condition: Both users must be logged in.
   Post-condition: Messages are accessible in the user's inbox.
   Exception: An error message is shown if the message cannot be sent.

7. **Notifications:**
   Description: Sends notifications to users about new events, applications, and updates.
   Input: Event triggers (new event, application status, updates).
   Source: System.
   Output: Notification message.

Destination: User (Organization and Volunteer).
Action: Notification is sent via email or in-app notification.
Pre-condition: User must have notifications enabled.
Post-condition: User receives and views the notification.
Exception: An error message is shown if the notification cannot be sent.

8. **Review Posting and Viewing:**
   Description: Users can post and view reviews for events and organizations.
   Input: Review text, rating, event/organization ID.
   Source: User (Volunteer).
   Output: Review is stored in the database and displayed on the event/organization's profile.
   Destination: Database.
   Action: Review is saved and can be viewed by other users.
   Pre-condition: User must be logged in and have participated in the event.
   Post-condition: Review is visible to all users.
   Exception: An error message is shown if any input is empty or does not meet validation criteria.

9. **Event Details Updation :**
   Description: Organizations can update event details.
   Input: Event ID, updated event details (name, description, date, time, location, number of volunteers needed, required skills).
   Source: Organization.
   Output: Updated event details are stored in the database.
   Destination: Database.
   Action: Event details are updated.
   Pre-condition: Organization must be logged in and must be the creator of the event.
   Post-condition: Updated event details are visible to volunteers.
   Exception: An error message is shown if the update action fails.

10. **Event Deletion:**
    Description: Organizations can delete event created by them.

Input: Event ID [selecting the event that needs to be deleted]
Source: Organization.
Output: event details are deleted from the database.
Destination: Database.
Action: Event is deleted
Pre-condition: Organization must be logged in and must be the creator of the event.
Post-condition: delete event are no longer visible to volunteers.
Exception: An error message is shown if the deletion action fails.

11. Profile Updation:

Description: Users can update their profile information and change their passwords.
Input: Updated profile information (name, phone, email, etc.), old password, new password.
Source: User (Organization and Volunteer).
Output: Updated profile information and password are stored in the database.
Destination: Database.
Action: Profile information and password are updated.
Pre-condition: User must be logged in.
Post-condition: Updated profile information and password are saved.
Exception: An error message is shown if the input is invalid or if the old password does not match the current password.

12. Profile Deletion:

Description: Delete the user's profile i.e details from the database permanently.
Input: Volunteer_ID (for Volunteer), Org_ID (for Organization)
Source: User (Organization and Volunteer).
Output: message informing the user whether their profile is deleted or not.
Destination: Database.
Action: the user info is deleted from database.
Pre-condition: User must be logged in.
Post-condition: User is Logged Out.
Exception: An error message is shown in case of failure.

13. **Forgot Password (OTP Verification):**

Description: Users can reset their passwords using OTP verification.
Input: Email, OTP, new password.
Source: User (Organization and Volunteer).
Output: OTP is sent to the user's email, new password is stored in the database.
Destination: User's email (for OTP), Database (for new password).
Action: OTP is sent to the user, and the user can reset their password using the OTP.
Pre-condition: User must provide a registered email.
Post-condition: Password is reset and the user can log in with the new password.
Exception: An error message is shown if the email is not registered or if the OTP is incorrect.

14. **Admin Control (Monitoring and Managing Events):**

Description: Admin can monitor and manage events created by the organization this involves approving or rejecting the events.
Input: Admin credentials, event details for management actions.
Source: Admin.
Output: Admin actions (approve/reject events)
Destination: Database.
Action: Admin performs management actions on events.
Pre-condition: Admin must be logged in with administrative privileges.
Post-condition: Admin actions are applied, and system data is updated accordingly.
Exception: An error message is shown if the admin action fails or if the admin does not have the necessary privileges.

15. **Posting Pictures and Comments**:
Description: Users (volunteers and organizations) can post pictures related to volunteering events and comment on pictures, similar to Instagram or Facebook.
Input: Picture file, caption, event ID (for posting pictures); comment text, picture ID (for commenting).
Source: User (Volunteer and Organization).
Output: Picture and caption are stored in the database; comments are stored and displayed on the picture.
Destination: Database.
Action: Pictures are uploaded and displayed on the event's page; comments are added and displayed below the picture.
Pre-condition: User must be logged in.
Post-condition: Picture and comments are visible to all users.
Exception: An error message is shown if the upload or comment fails due to invalid input or system errors.

16. **Change Password**:
Description: Users can change their password.
Input: Old Password, New Password.
Source: User (Volunteer and Organization).
Output: Message informing the user whether password is changed or not.
Destination: Database.
Action: the old password is replaced with the new one.
Pre-condition: User must be logged in.
Post-condition: None
Exception: An error message is shown if the old password doesn't match the current password from the database.

17. Admin Control (Monitoring and Managing Users):
Description: Admin can suspend and unsuspend user's accounts (volunteers and organizations).
Input: Admin credentials, user's details for management actions.
Source: Admin.
Output: Admin actions (activate/deactivate user accounts)
Destination: Database.

Action: Admin performs management actions on user accounts.
Pre-condition: Admin must be logged in with administrative privileges.
Post-condition: Admin actions are applied, and system data is updated accordingly.
Exception: An error message is shown if the admin action fails or if the admin does not have the necessary privileges.

Chapter 4: System Design

4.1 Module Diagram:

Module diagram is a diagram which is used for showing the allocation of classes and objects to module in the physical design of a system. Module Diagram indicates the partitioning of the system architecture. Through this diagram, it is possible to understand the general physical architecture of a system. The two essential elements of a module diagram are modules and their dependencies.

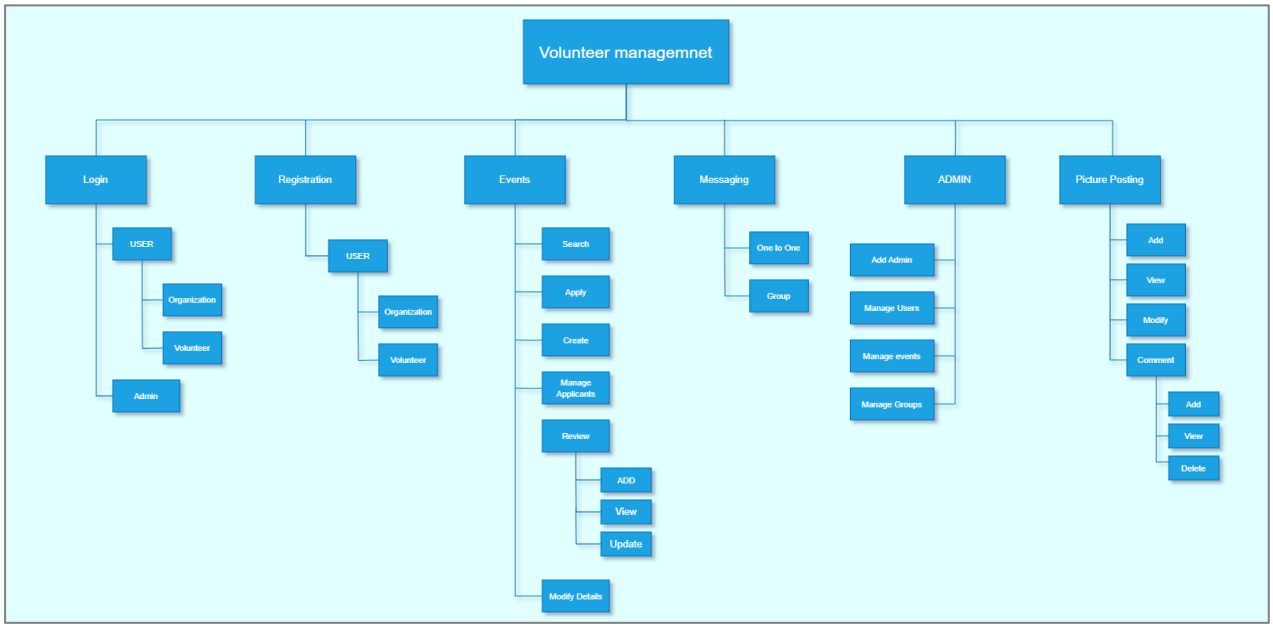
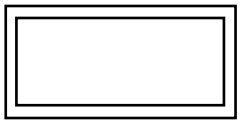
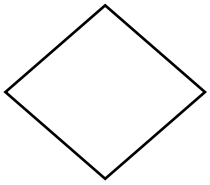
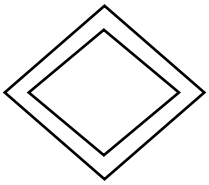
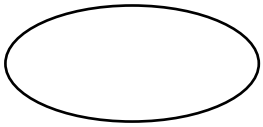
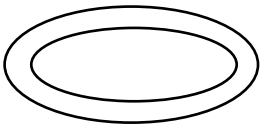
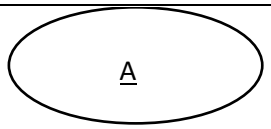


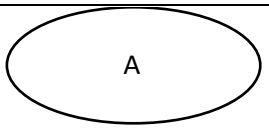
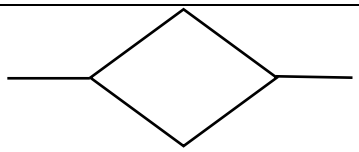
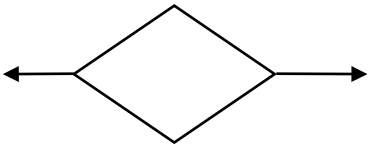
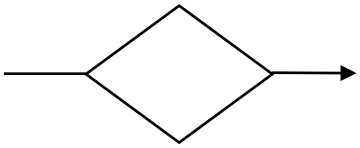
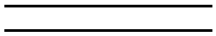
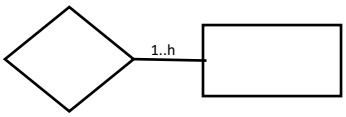
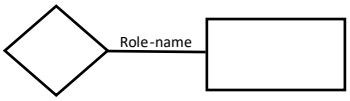
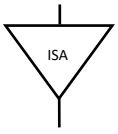
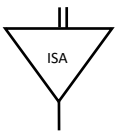
Figure 4.1 Module Diagram

4.2 Entity Relationship Diagram:

An Entity-Relationship Diagram is a visual representation used to model the structure of a database. It illustrates the relationships between different entities, which represent real-world objects, concepts, or information within the database system. ER diagrams consist of entities, attributes which are the properties of those entities, relationships which are the connections between those entities, and cardinality among those entities. ER diagrams are essential tools for database design, helping developers and stakeholders understand the logical organization of data, identifying key relationships, and ensuring the proper implementation of a database schema. By providing a clear and intuitive overview, ER diagrams play a crucial role in improving communication and ensuring the accuracy of database systems in various domains, such as business, software development, and data management. ER model becomes an abstract data model that defines a data or information structure which can be implemented typically in a relational database.

Diagram Notations:

Name	Symbol	Description
Rectangle		Entity set
Double Rectangle		Weak Entity set
Diamond		Relationship set
Double Diamond		Identifying relationship set for weak entity set
Eclipse		Attribute
Double Eclipse		Multivalued attribute
Dotted Eclipse		Derived attribute
Primary key		Primary key

Discriminator		Discriminating attribute of weak entity set
Many-to-many		Many-to-many relationship
One-to-one		One-to-one relationship
Many-to-one		Many-to-one relationship
Double line		Total participation of entity set in a relationship
Line		Links attribute to entity set or represents Partial participation of entity set in a relationship.
Mapping cardinality		Cardinality limits
Role indicator		Role indicator
ISA		ISA (specialization or generalization)
Total generalization		Total generalization

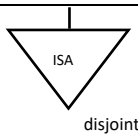
Disjoint generalization		Disjoint generalization

Table 4.1 Entity Relationship Diagram Symbols

4.2.1 Entity Sets:

1. Users:

It stores information about all users (volunteers and organizations) registered on the platform.

Attributes:

i. **User_id** – single valued, simple
   - It uniquely identifies each user.

ii. **Name** – single valued, simple
   - It stores the name of the user.

iii. **Email** – single valued, simple
   - It stores the email address of the user.

iv. **Phone** – single valued, simple
   - It stores the phone number of the user.

v. **Password** – single valued, simple
   - It stores the encrypted password of the user.

vi. **User_type** – single valued, simple
   - It indicates whether the user is a volunteer or an organization.

vii. **Profile_Picture** – single valued, simple
   - It stores the profile picture of the user.

viii. **Address** – single valued, composite
   - It stores the address of the user.

ix. **Registration_date** – single valued, simple
   - It stores the date when the user registered.

x. **DOB/Date of establishment** – single valued, simple
   - It stores the date of birth for volunteer's and date of establishment for organisation.

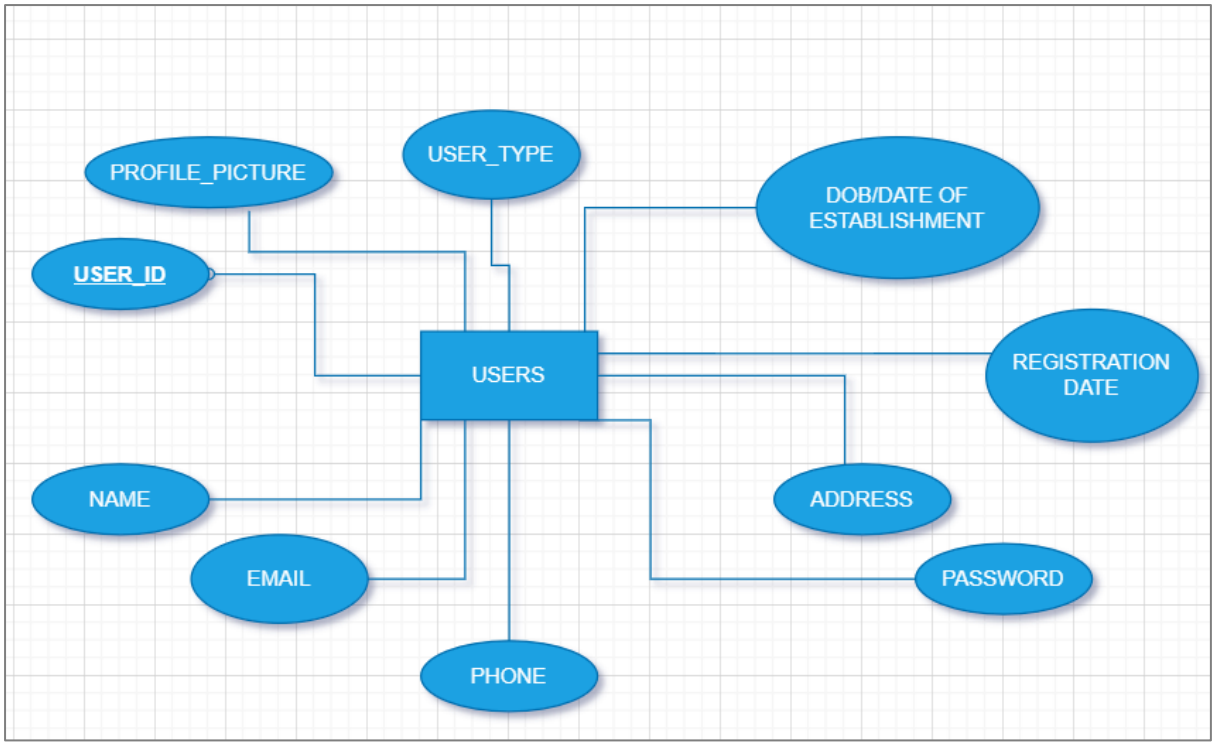


Figure 4.2 Users Entity Set

2. Events:

It stores information about all events posted by organizations.

Attributes:

i. **event_id** – single valued, simple
   - It uniquely identifies each event.

ii. **event_name** – single valued, simple
   - It stores the name of the event.

iii. **description** – single valued, simple
   - It stores the description of the event.

iv. **From_date** – single valued, simple
   - It stores the start date of the event.

v. **To_date** – single valued, simple
   - It stores the end date of the event.

vi. **From_time** – single valued, simple
   - It stores the start time of the event.

vii. **To_time** – single valued, simple
   - It stores the end time of the event.

viii. **location** – single valued, composite
   - It stores the location of the event.

ix. **volunteers_needed** – single valued, simple

- It stores the number of volunteers needed for the event.

x. **Max_Applicantions** – single valued, simple
   - it stores the maximum number of application that a event can have.

xi. **status** – single valued, simple
   - It indicates the status of the event (e.g., active, completed, cancelled).

xii. **Date_of_Creation** – single valued, Simple
   - It indicates the date on which the event was created.

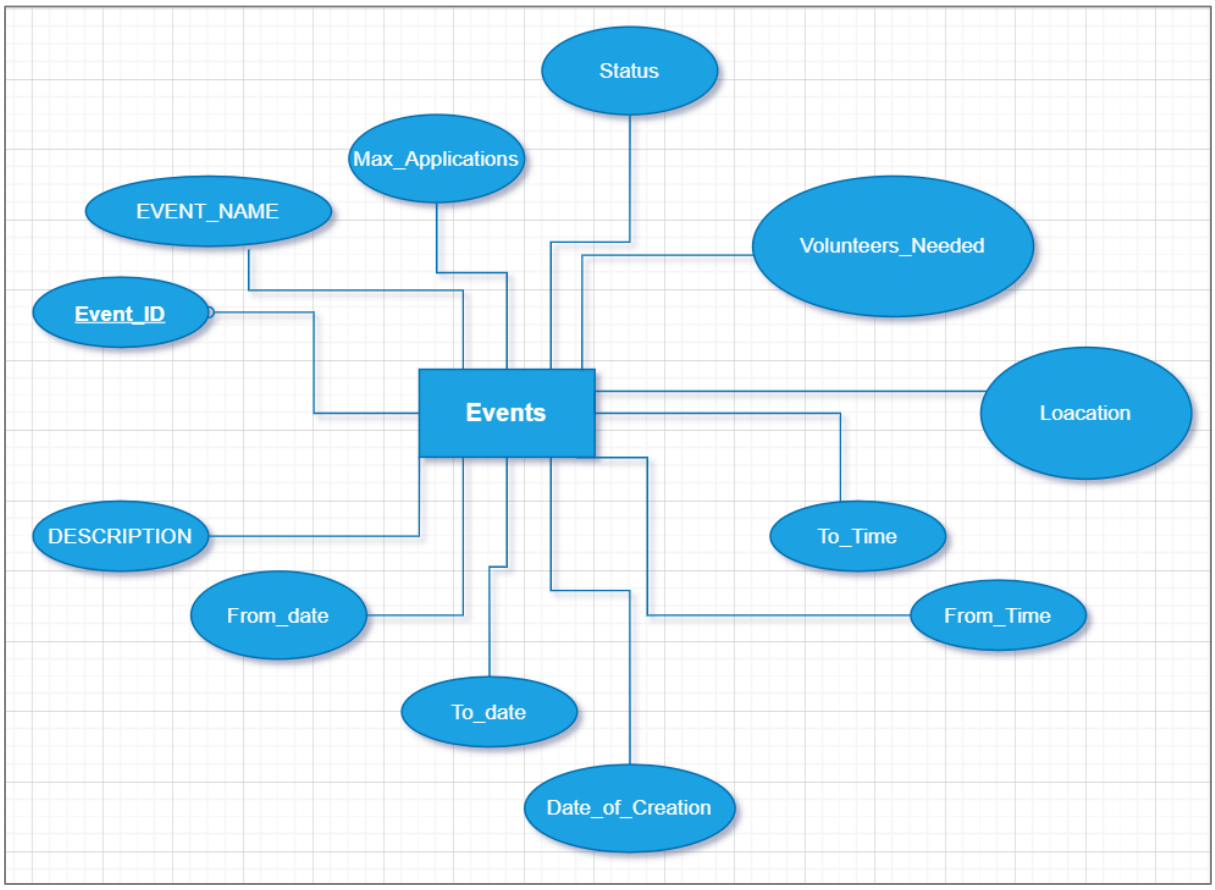


Figure 4.3 Events Entity Set

3. Feedback_Rating:

It stores the review and rating given by the users for an event.

Attributes:

i. **review_id** – single valued, simple
   - It uniquely identifies each review.

ii. **Description** – single valued, simple
   - it stores the description of the review.

iii. **Date**- single valued, simple

- It stores the date of review.

iv. **Rating** – single values, simple
- it stores the rating given by the users.

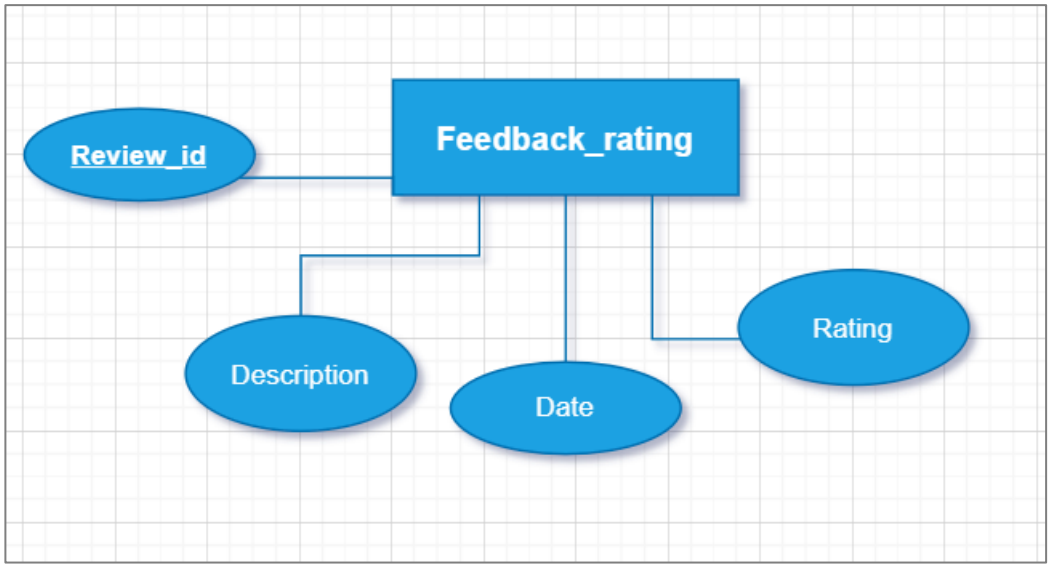


Figure 4.4 Feedback_Rating Entity Set

4. Post_Picture

It stores pictures uploaded by users related to events.

Attributes:

i. **picture_id** – single valued, simple
- It uniquely identifies each picture.

ii. **picture_url** – single valued, simple
- It stores the URL of the picture.

iii. **caption** – single valued, simple
- It stores the caption of the picture.

iv. **upload_date** – single valued, simple
- It stores the date when the picture was uploaded

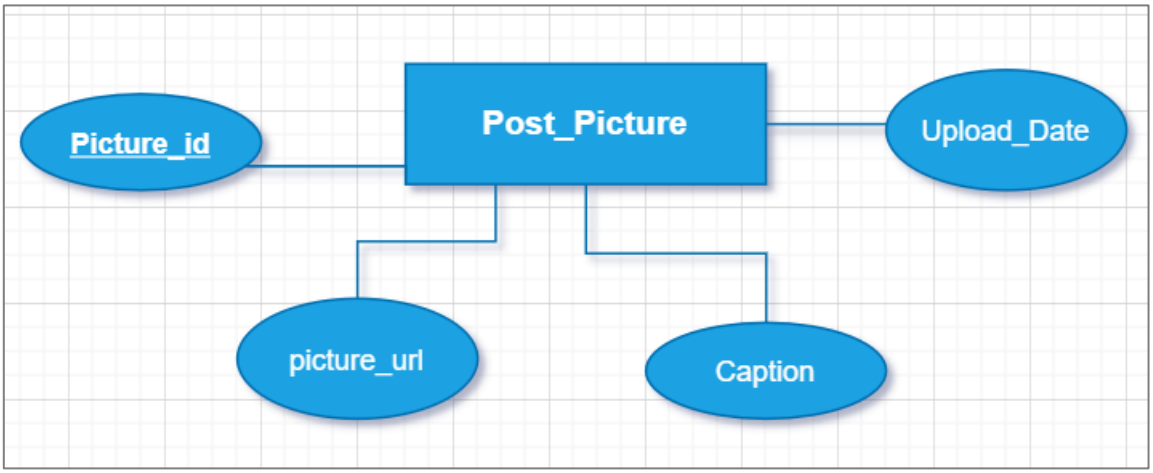


Figure 4.5 Post_Picture Entity Set

5. Comment:

It stores comments posted by users on pictures.

Attributes:

i. **comment_id** – single valued, simple
   - It uniquely identifies each comment.

ii. **comment_text** – single valued, simple
   - It stores the text of the comment.

iii. **date** – single valued, simple
   - It stores the date when the comment was posted.

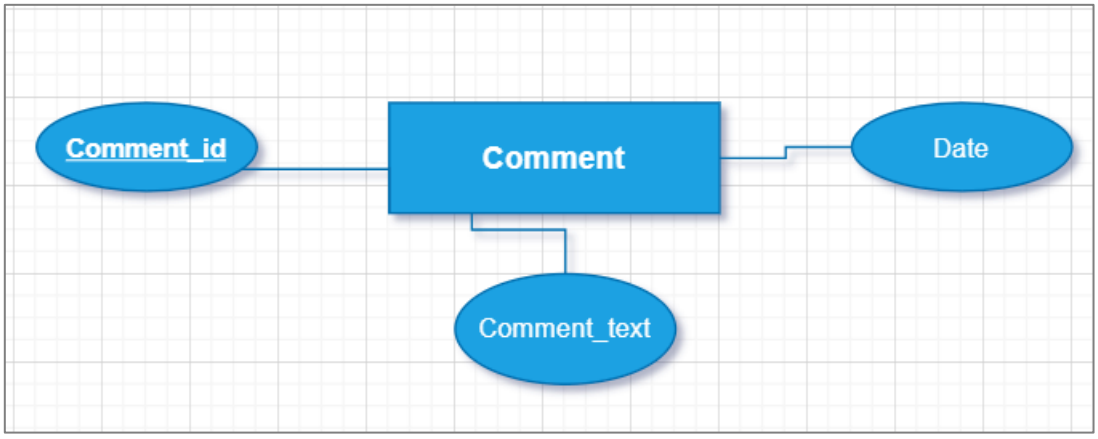


Figure 4.6 Comment Entity Set

6. Causes:

It stores the causes for what the organisation and volunteers are working for.

Attributes:

i. **Cause_id** – single valued, simple
   - It uniquely identifies the causes.

ii. **Cause_name** – single valued, simple
   - It stores the name of the causes.

Figure 4.7 Cause Entity Set

7. Organization_type:

It stores the type of organization.

Attributes:

i. **Organization_id** – single valued, simple
   - It uniquely identifies each organization type.

ii. **Type** – single valued, simple
   - It stores the type name of the organization.

Figure 4.8 Organization_Type Entity Set

8. Skill:

It stores the information about the skill that volunteers can have and that organization might require for events

Attributes:

i. **skill_id** – single valued, simple
   - It uniquely identifies each skill.

ii. **skill_name** – single valued, simple

- It stores the name of the skill.

Figure 4.9 Skill Entity Set

9. Message

It stores messages exchanged between users .

Attributes:

i. **Message_id** – single valued, simple
   - it uniquely identifies the messages.

ii. **Content** – single valued, simple
   - it stores the content or text of the message.

iii. **Timestamp** – single valued, simple
   - Stores the time and date when the message was sent.

iv. **Image_url** – single valued, simple
   - Stores the url of the image if sent as a message.

v. **Status** – single valued, simple
   - it indicates the status of a message (sent, read etc)

vi. **Message_category** – single valued, simple
   - It indicates the message type i.e (individual) for one on one messaging and (group) for group messaging.

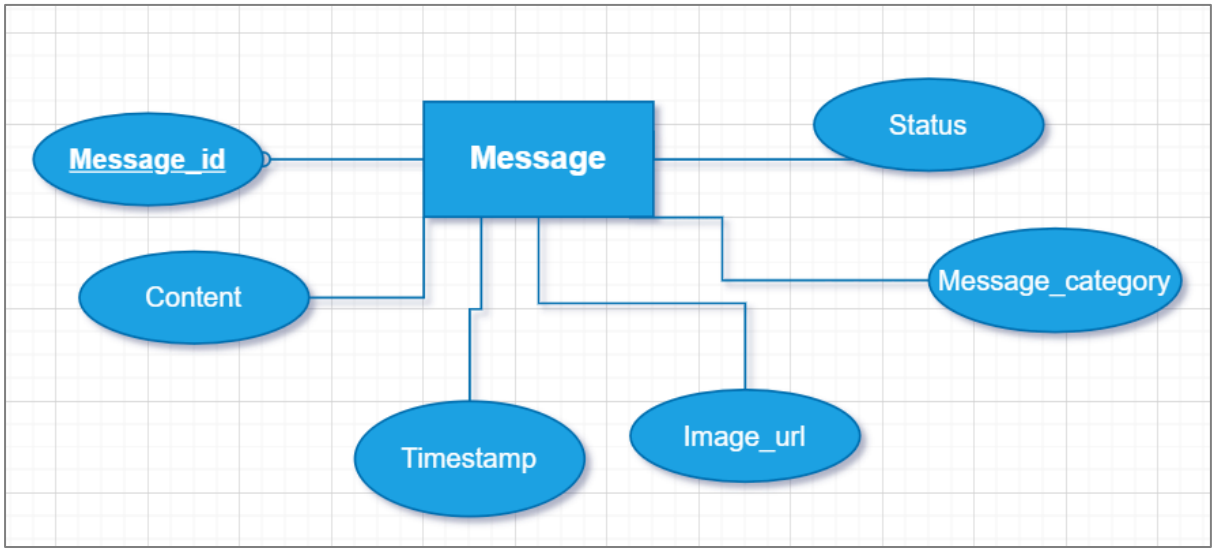


Figure 4.10 Message Entity Set

10. Group:

It stores the group data created by the user for communication.

Attribute:

i. **Group_ID** – single valued, simple
   - it uniquely identifies each group.

ii. **Group_name** – single valued, simple
   - It stores the name of the group.

iii. **Date_of_Creation** – single valued, simple
   - It stores the date on which the group was created.

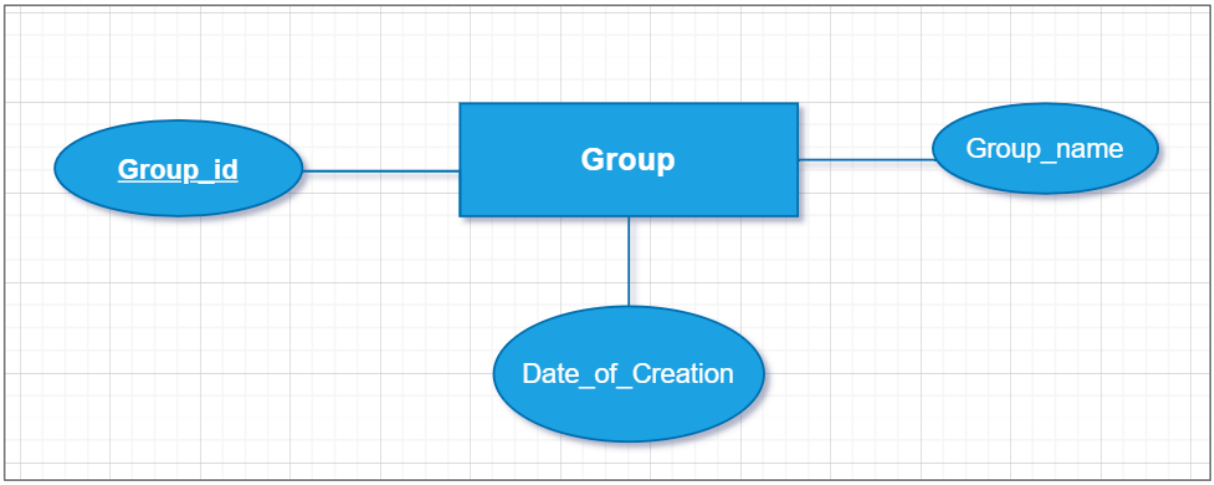


Figure 4.11 Group Entity Set

11. Admin:

It stores information about the admin's who manage the system.

Attribute:

i. **Admin_id** – single valued, simple
   - it uniquely identifies the admin of the system.

ii. **Full_name** – single valued, simple
   - It stores the name of the admin.

iii. **Email** – single valued, simple
   - It stores the email of the admin

iv. **Password** – single valued, simple
   - It stores the encrypted password of the admin

v. **Role** – single valued, simple
   - It stores the role of the admin's (eg super admin, event manager etc)

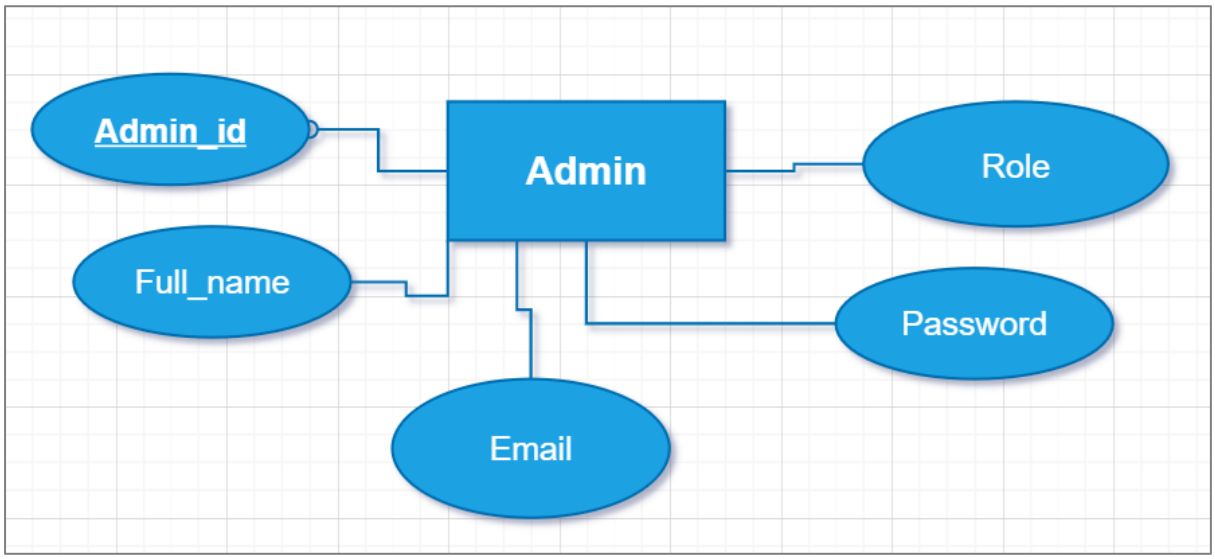


Figure 4.12 Admin Entity Set

4.2.2 Relationship Sets:

1. **User-->WorksFor-->Causes:**
   - It specifies the causes, a user is working for.
   - Many to Many.
   - Total Participation from Users.

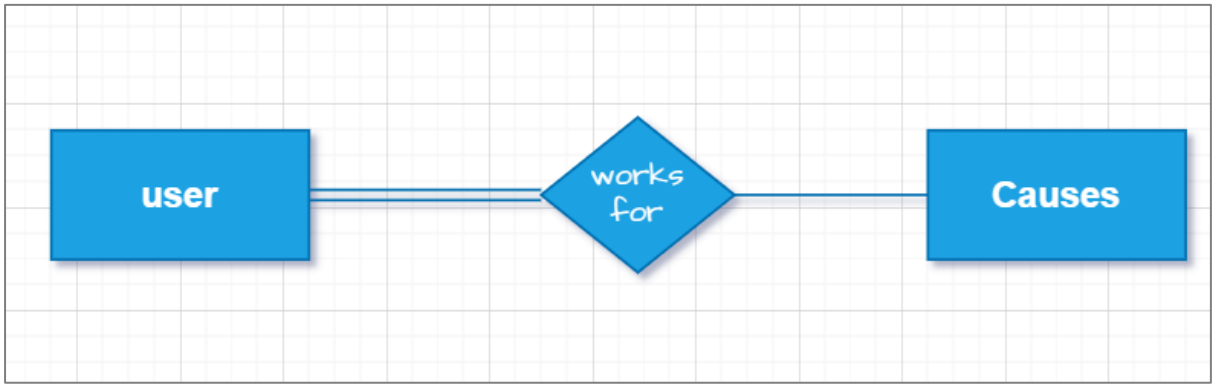


Figure 4.13 WorksFor Relationship Set

2. **User-->has-->Skills:**
   - It specifies the skill a user (Volunteer) has.
   - Many to Many.
   - Partial Participation Throughout.

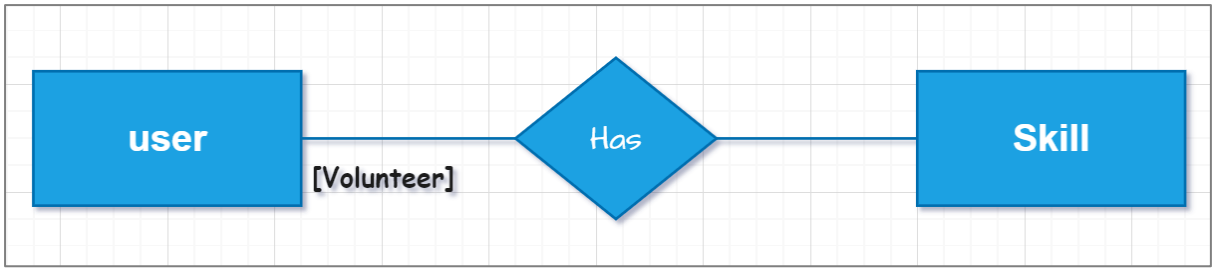


Figure 4.14 Has Relationship Set

3. **User-->Creates-->Events:**
    - It specifies which user (organisation) created which event .
    - One to Many.
    - Total Participation from Events.

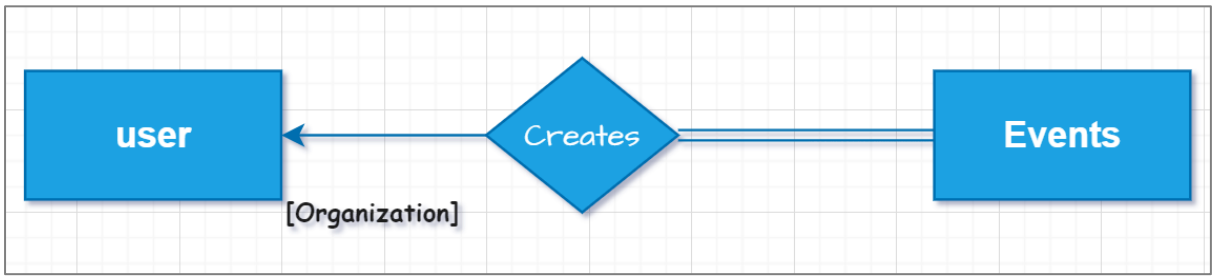


Figure 4.15 Creates Relationship Set

4. **User-->ApplyFor-->Events:**
    - It specifies which user (volunteer) applied to which volunteering event.
    - Many to Many.
    - Partial Participation Throughout.
    - Discriptive attribute:
        i. **Date** – single valued, simple
            - It stores the date of which the application was made.
        ii. **Status** – single valued, simple
            - It specifies the status of the application (pending, accepted, rejected).

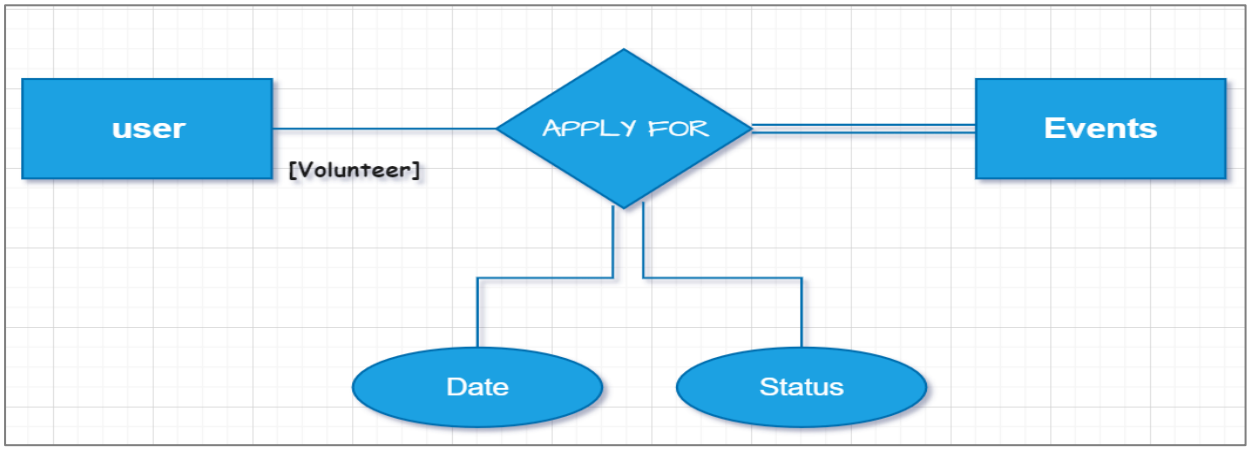


Figure 4.16 ApplyFor Relationship Set

5. **Events-->Require-->Skill:**
    - It stores the skill that are required by an event.
    - Many to Many.
    - Total Participation from Events.

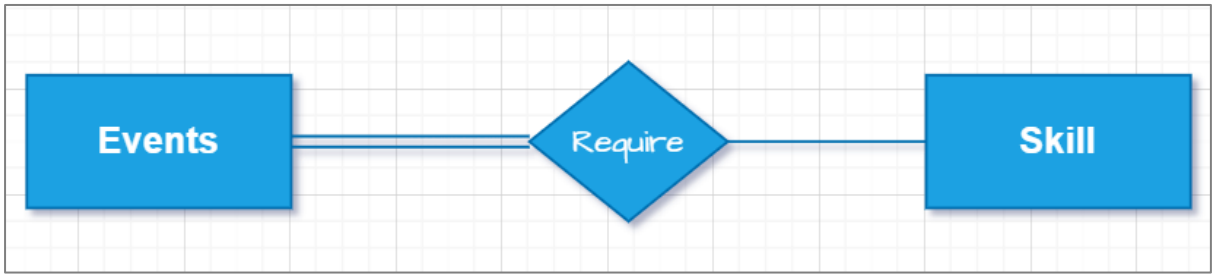


Figure 4.17 Require Relationship Set

6. **User-->Gives--> Rating & Feedback-->For-->Events :**
    - It stores the rating and feedback provided by a user to an event.
    - One to Many.
    - Total Participation from Rating & Feedback.

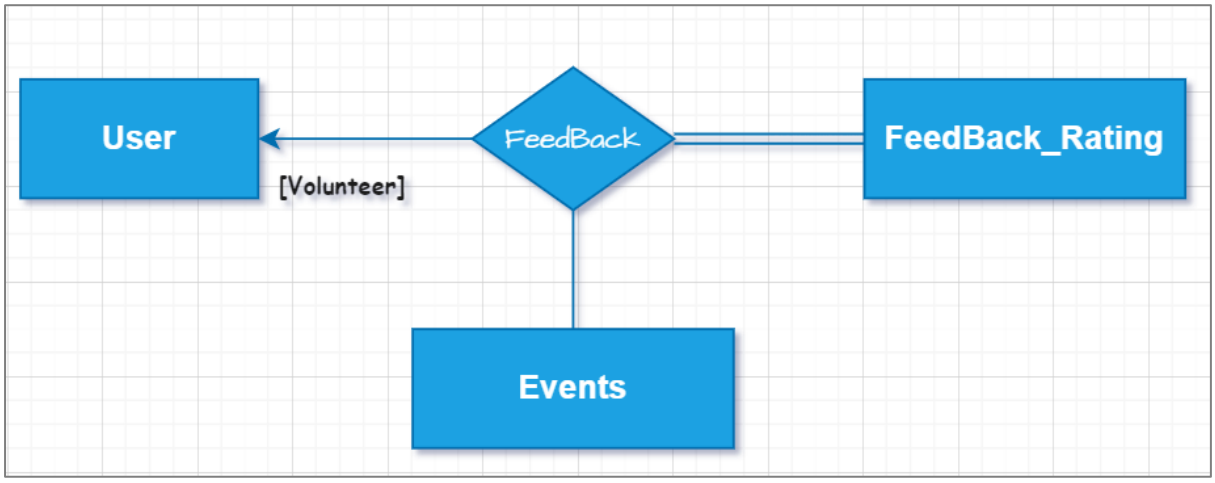


Figure 4.18 FeedBack Relationship Set

7. **User-->Post-->Post_Picture:**
    - It stores which image was posted by which user.
    - One to Many.
    - Total Participation from Post_Picture.

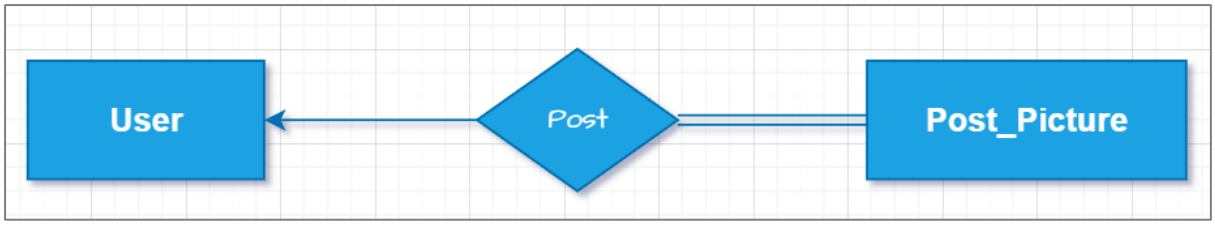


Figure 4.19 Post Relationship Set

8. **Post_Picture-->BelongsTo-->Events :**
    - It specifies which picture belongs to which events.
    - Many to One.
    - Partial Participation Throughout.

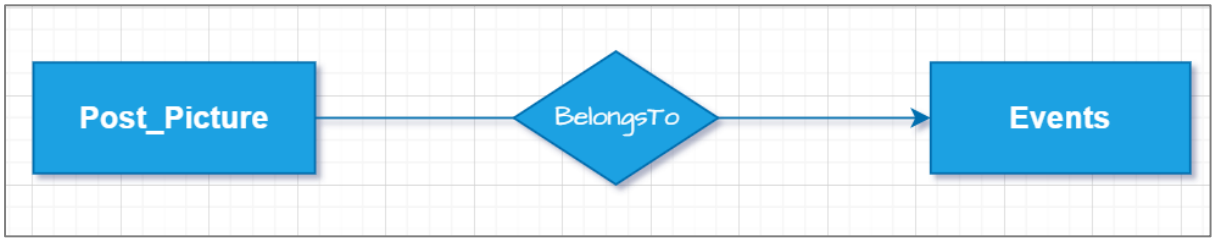


Figure 4.20 BelongsTo Relationship Set

9. **User-->Gives--> Comments-->On-->Post_Picture :**
    - It specifies which comment was done by which user on which Post_Picture.
    - One to Many.
    - Total Participation from Comments.

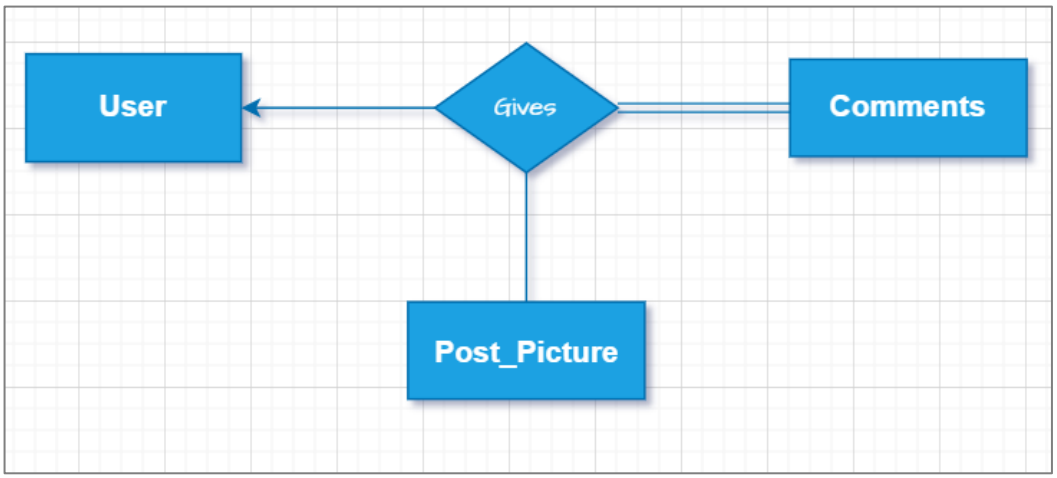


Figure 4.21 Gives Relationship Set

10. **User-->Join-->Group :**
    - It specifies which user Joined which group.
    - Many to Many.

- Total Participation from Group.
- Discriptive attribute:
    i. **Date_of_Joining** – single valued, simple
        - It stores the date of which the user Joins the Group.

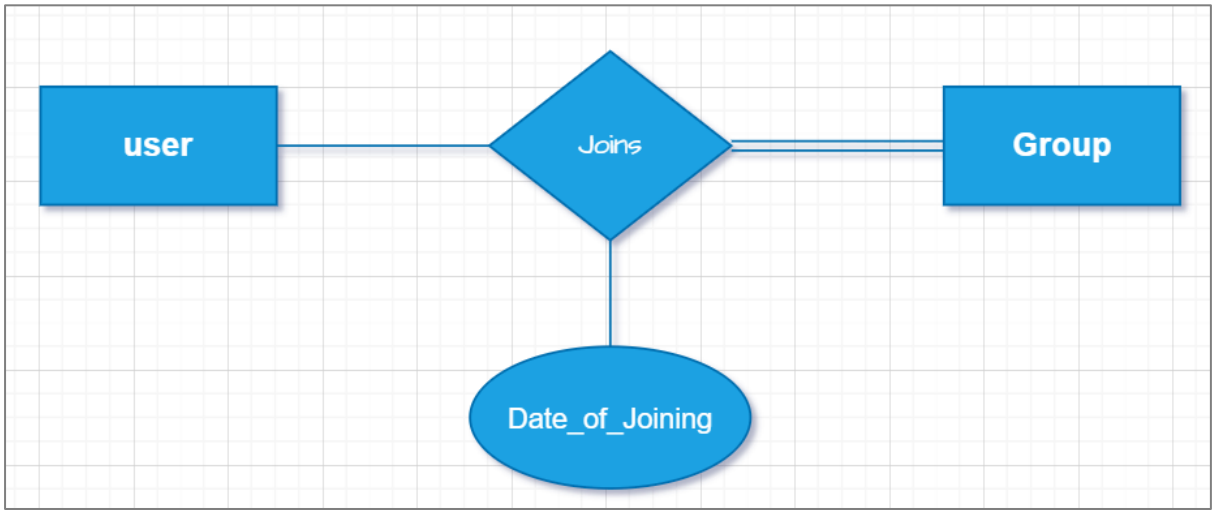


Figure 4.22 Joins Relationship Set

11. User-->Creates-->Group:

- It specifies which group was created by which user.
- One to Many.
- Total Participation from Group.
- Discriptive attribute:
    i. **Date_of_Creation** – single valued, simple
        - It stores the date of which the user Created the Group.

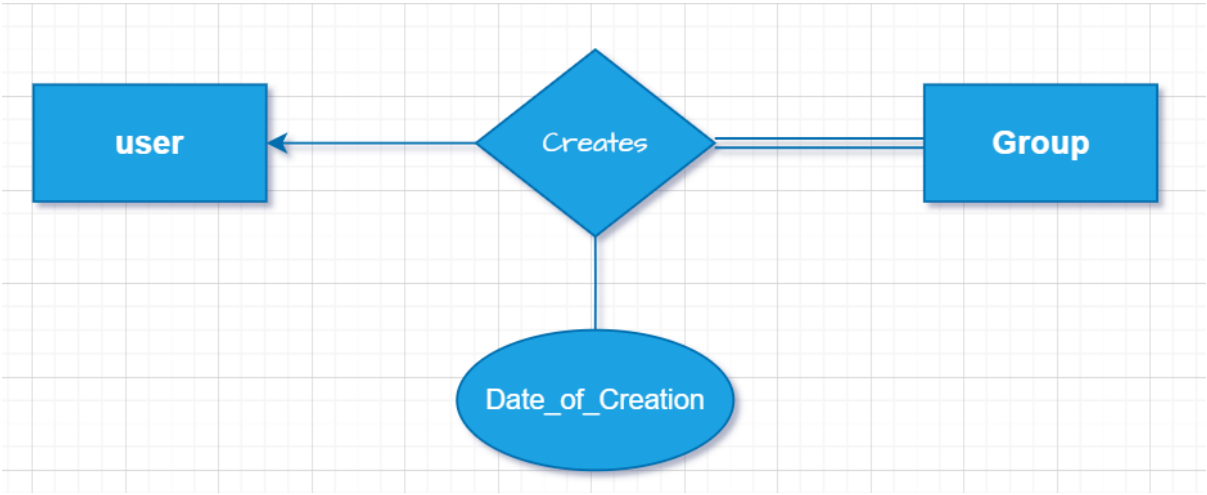


Figure 4.23 Creates Relationship Set

12. Group-->Has-->Message:

- It specifies the list of messages a group has.

- One to Many.
- Partial Participation throughout.

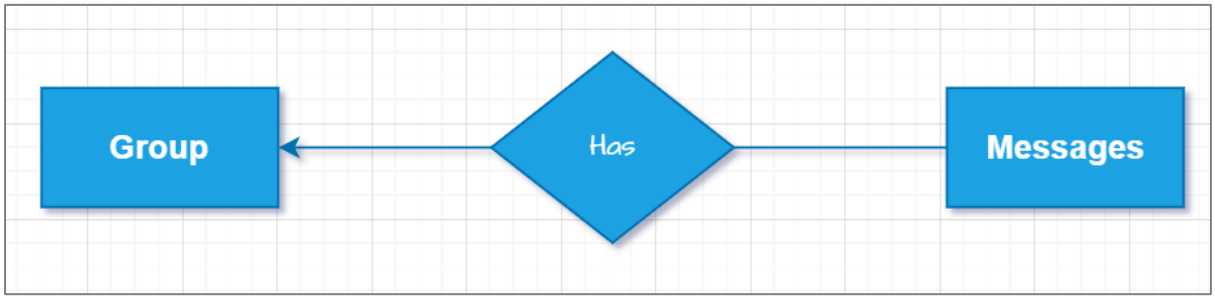


Figure 4.24 Has Relationship Set

13. User-->Sent-->Message:

- It specifies which message was sent by which user.
- One to Many.
- Total Participation from Message.

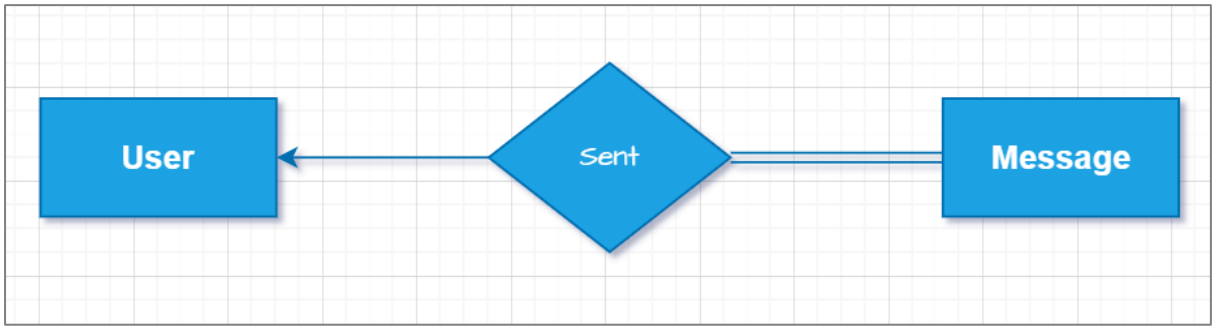


Figure 4.25 Sent Relationship Set

14. User-->Receive-->Message:

- It specifies which message was received by which user.
- One to Many.
- Total Participation from Message.

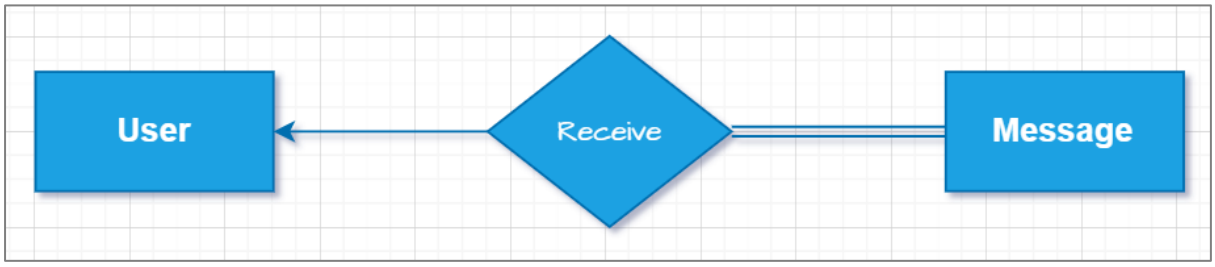


Figure 4.26 Receive Relationship Set

15. Admin-->Manages-->User:

- It specifies which user was suspended or unsuspended on which date by the admin.
- One to Many.
- Partial Participation Throughout.
- Discriptive attribute:
    i. **Date**– single valued, simple
        - It stores the date of which the user was suspended/unsuspended.
    ii. **Action** – single valued, simple
        - It stores the action (suspended/unsuspended)

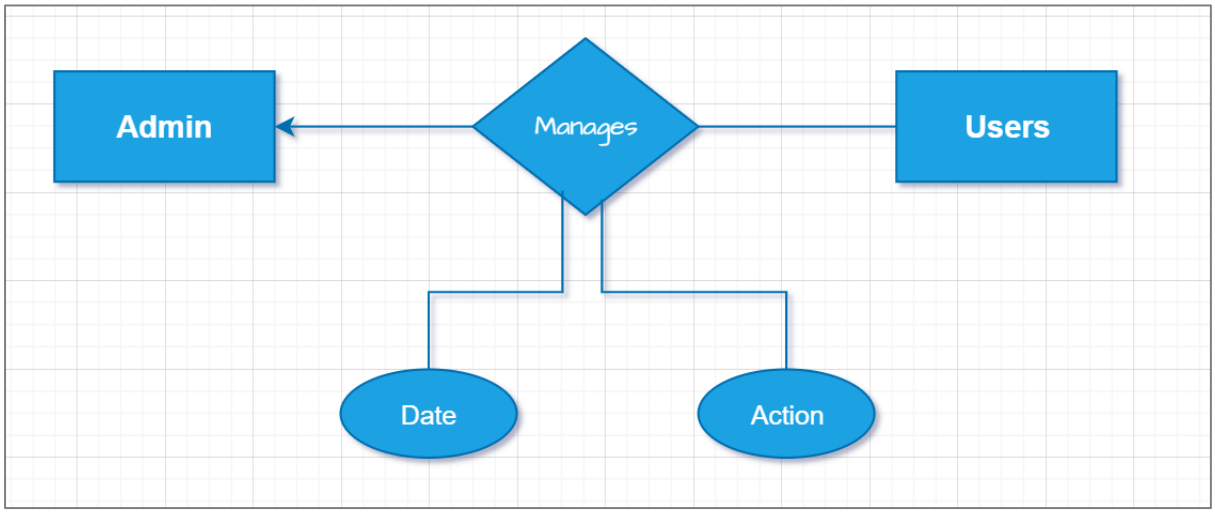


Figure 4.27 Manage Relationship Set

16. Admin-->Manages-->Events:

- It specifies which events was suspended or unsuspended on which date by the admin.
- One to Many.
- Partial Participation Throughout.
- Discriptive attribute:
    i. **Date**– single valued, simple
        - It stores the date of which the user was suspended/unsuspended.
    ii. **Action** – single valued, simple
        - It stores the action (suspended/unsuspended)

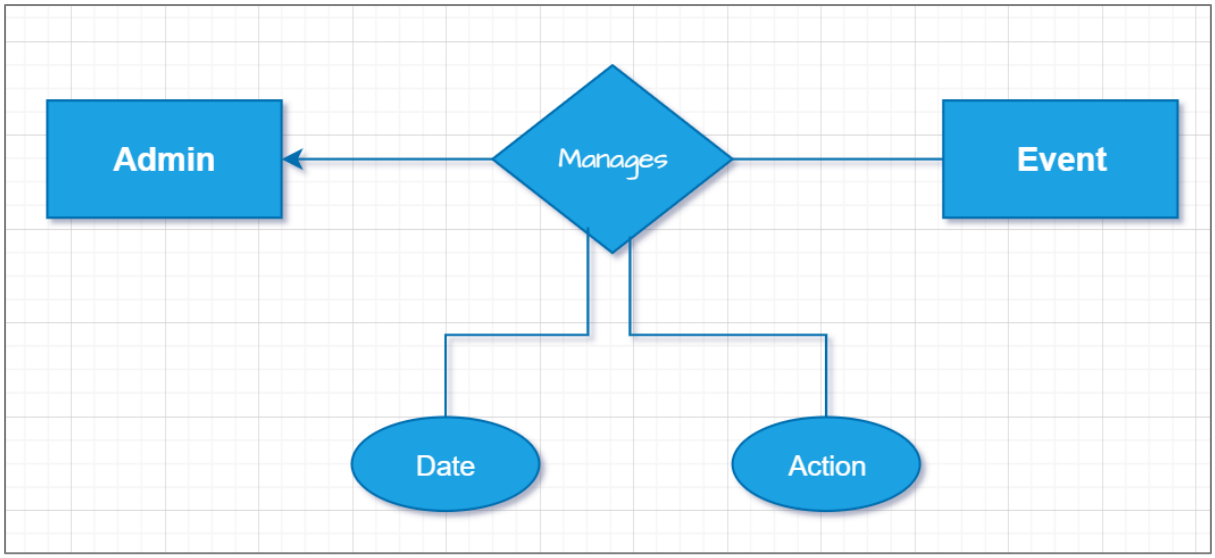


Figure 4.28 Manage Relationship Set

17. Admin-->Manages-->Groups:

- It specifies which Group was suspended or unsuspended by the Admin.
- One to Many.
- Partial Participation Throughout.
- Discriptive attribute:
  i. **Date**– single valued, simple
     - It stores the date of which the user was suspended/unsuspended.
  ii. **Action** – single valued, simple
     - It stores the action (suspended/unsuspended)

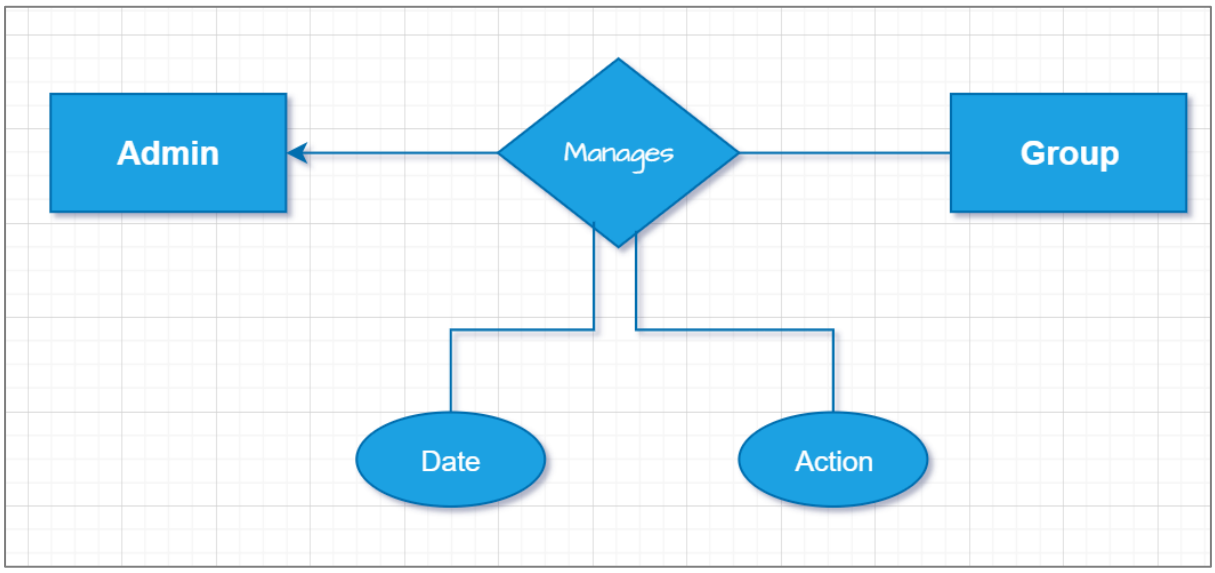


Figure 4.29 Manage Relationship Set

4.2.3 ER Diagram:

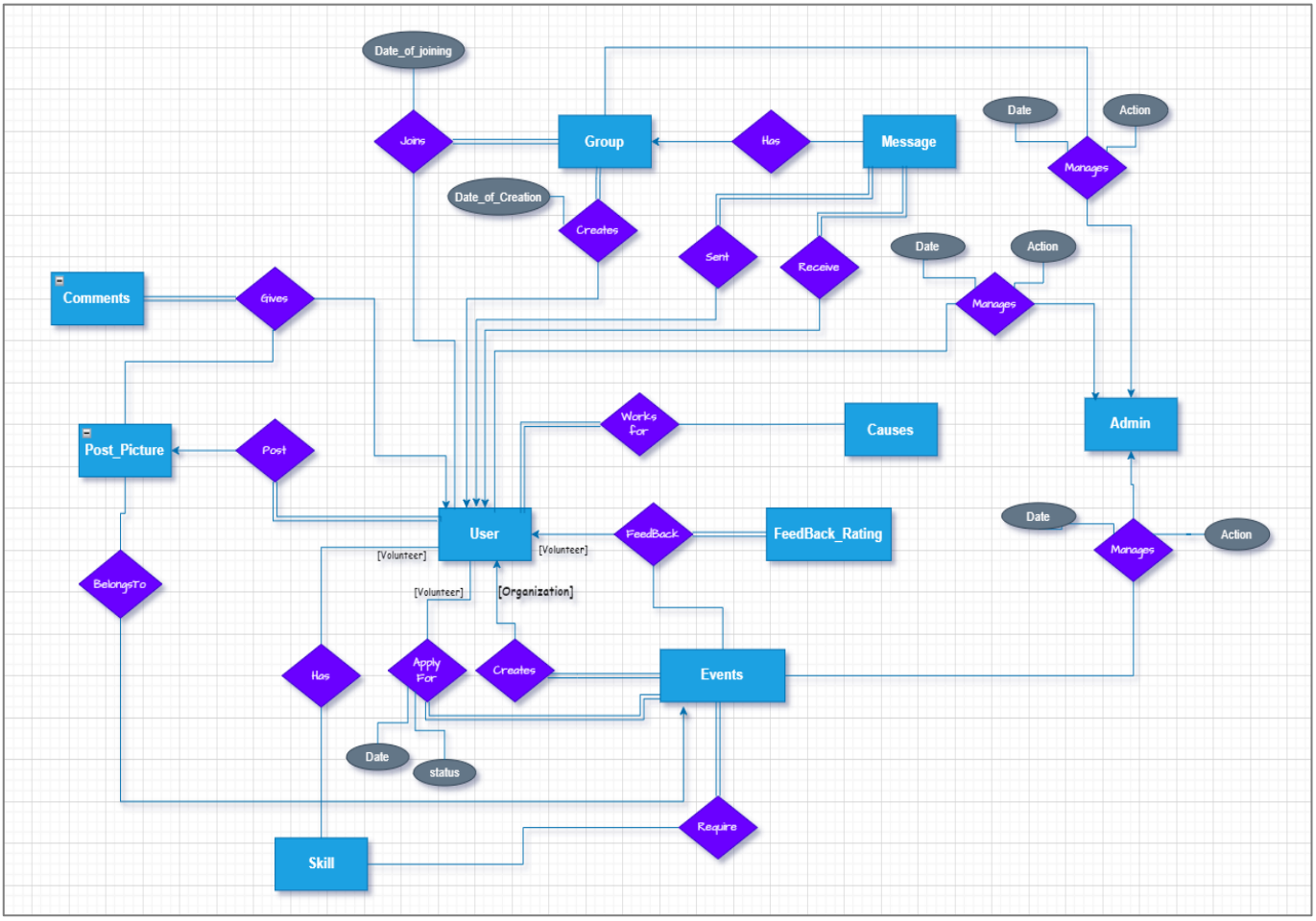


Figure 4.30 ER Diagram

4.3 Schema Diagram:

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organised and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

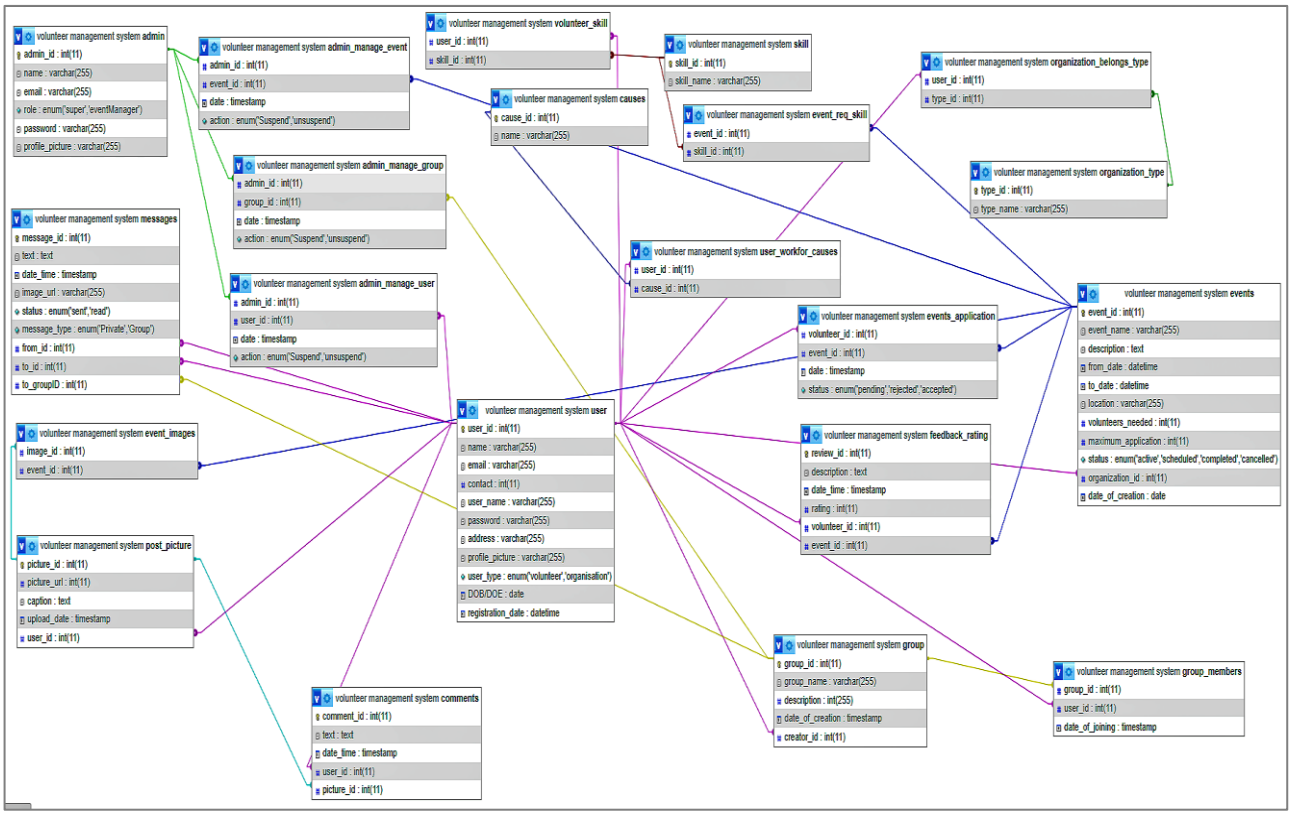
Diagram Notations:

Name	Symbol	Description
Table		A table is a collection of related data held in table format within a database.

Relation	→	In a relational database system, a one-to-one table relationship links two tables based on a Primary Key column in the child which is also a Foreign Key referencing the Primary Key of the parent table row. Therefore, we can say that the child table share the Primary Key with the parent table.

Table 4.2 Schema Diagram Notations

4.3.1 Schema Diagram:



Figures 4.31 Schema Diagram

4.4 Data Flow Diagram:

Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation. Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.

Diagram Notations:-

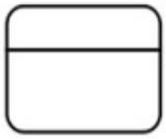
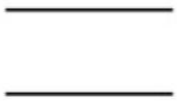
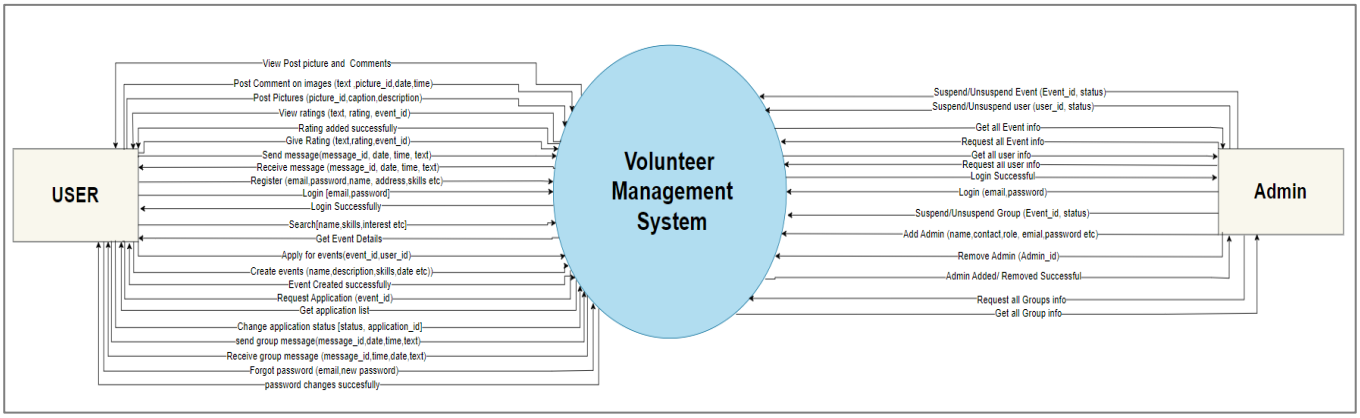
Name	Symbol	Description
Process		A process transforms incoming data flow into outgoing data flow.
Database		Data stores are repositories of data in the system.
Data Flow		Data flows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it.
External Entity		External entities are objects outside the system, with which the system communicates

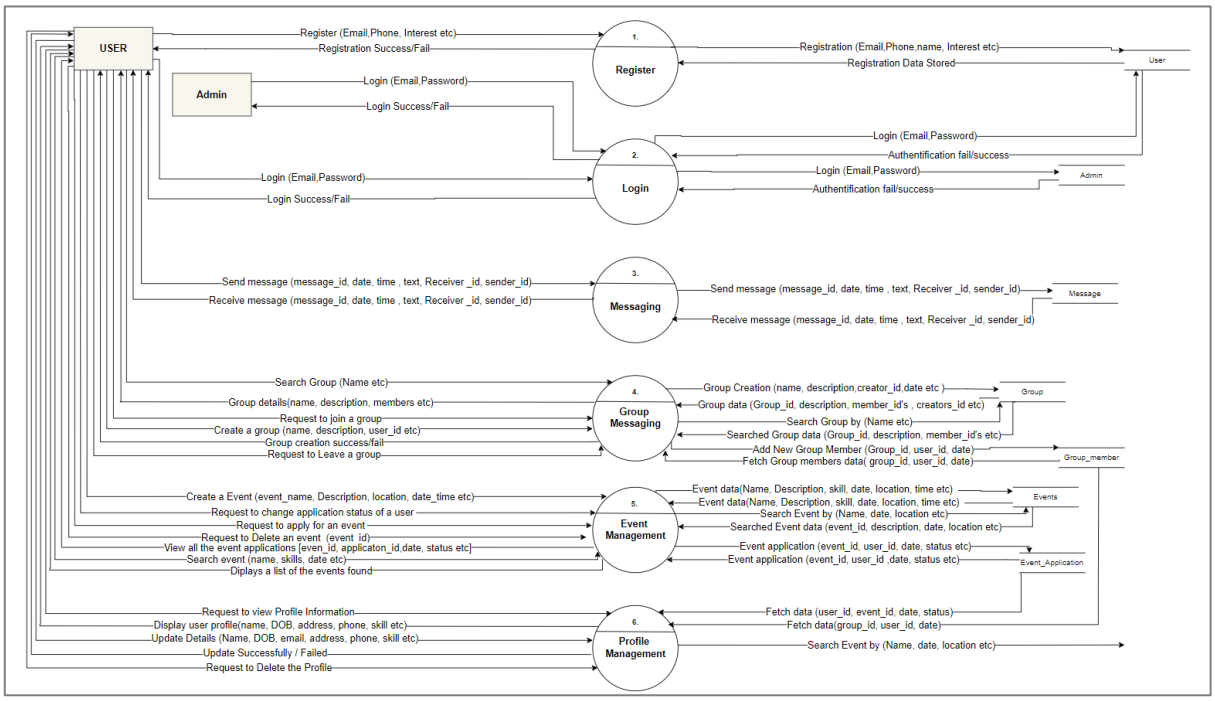
Table 4.3 Data Flow Diagram Notation

4.4.1 Level 0 (Context Level DFD):

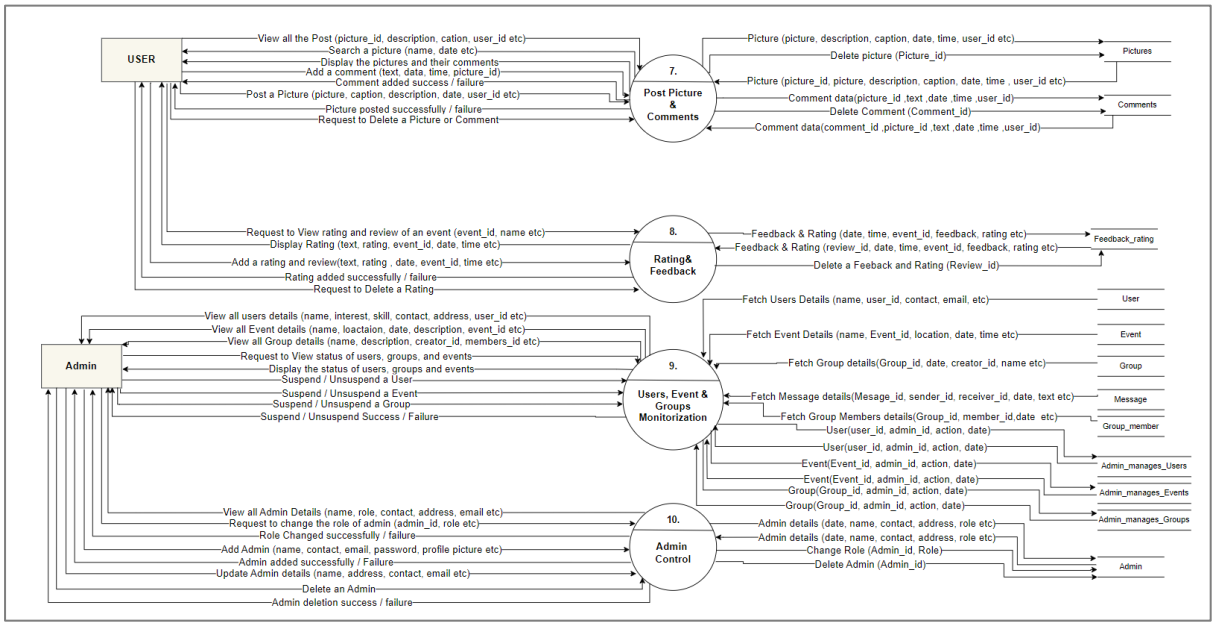


Figures 4.32 DFD Level – 0 (Context Level)

4.4.2 First Level DFD:

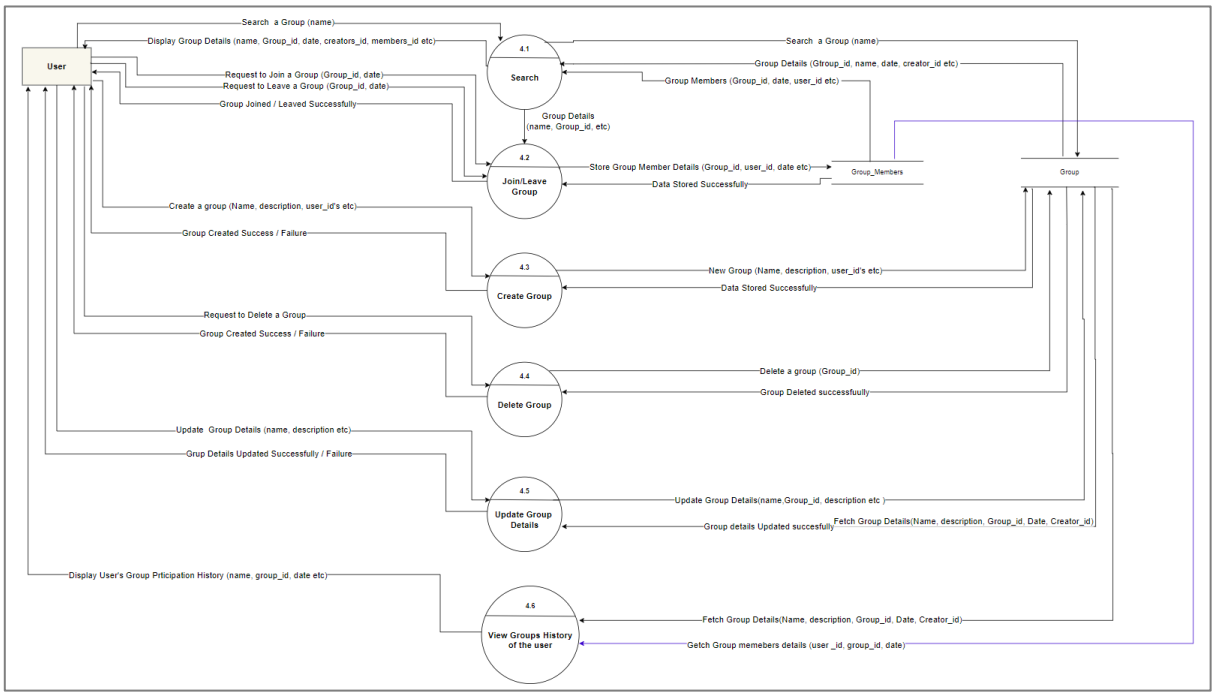


Figures 4.33 DFD Level – 1

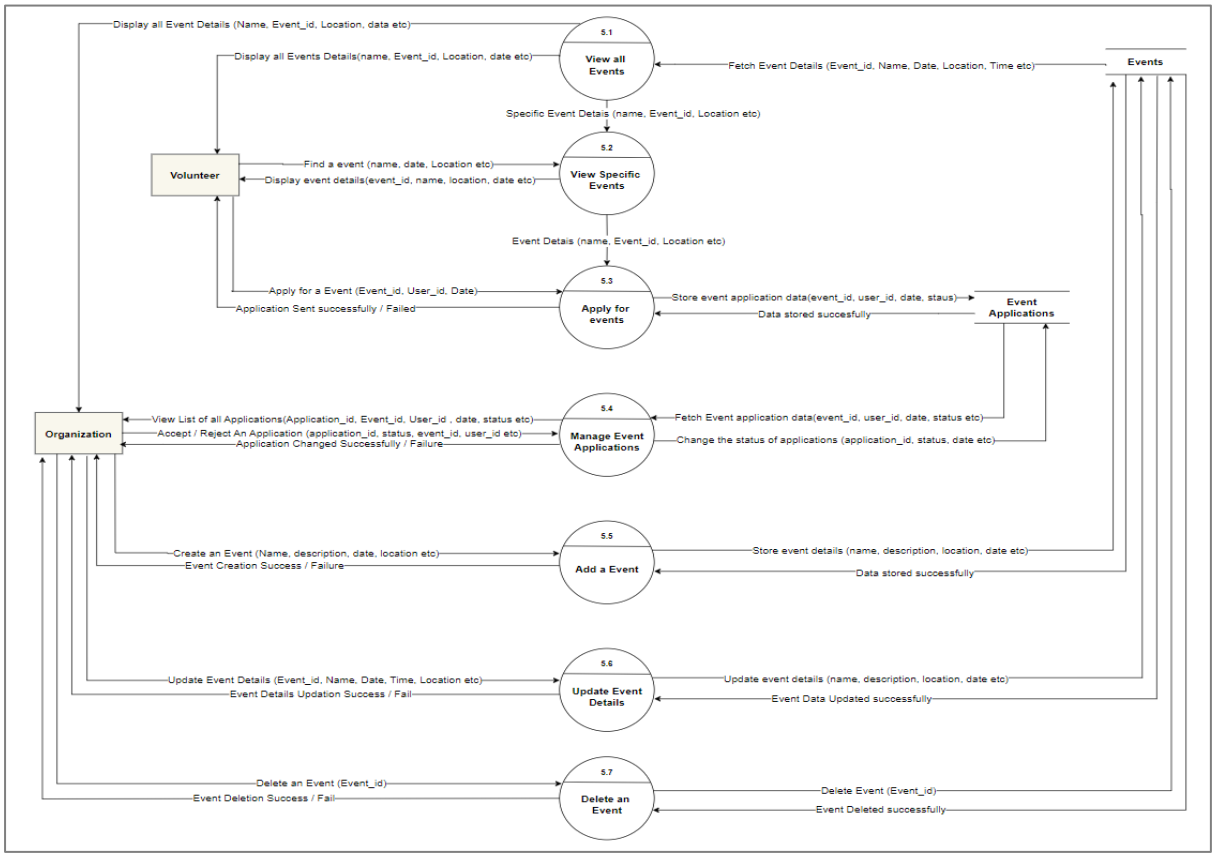


Figures 4.34 DFD Level – 1

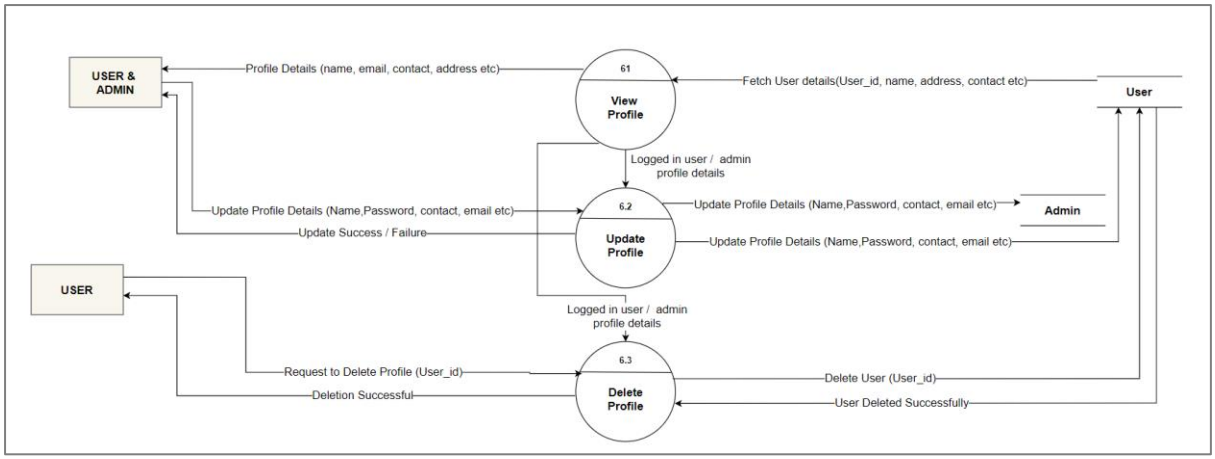
4.4.3 Second Level DFD:



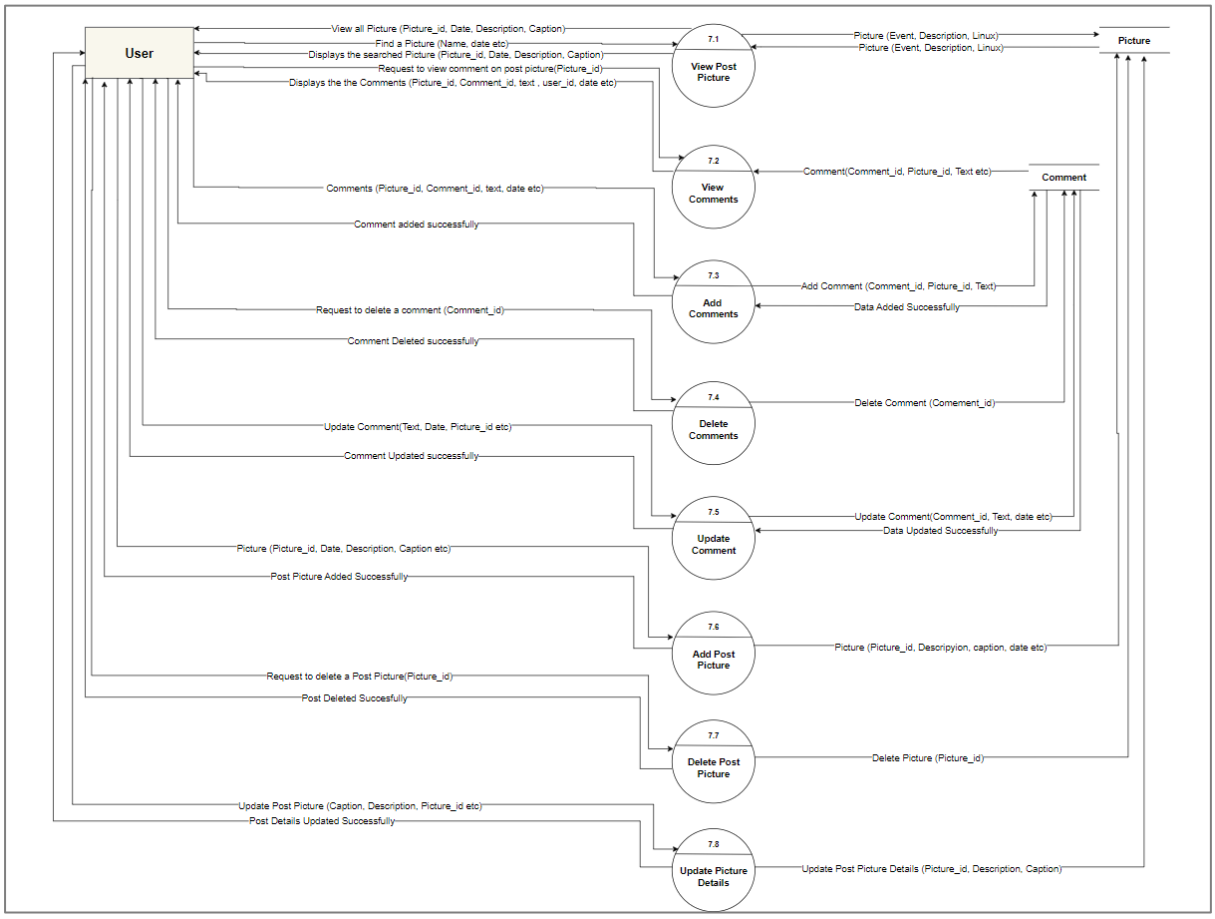
Figures 4.35 DFD Level – 2 for Group Messaging



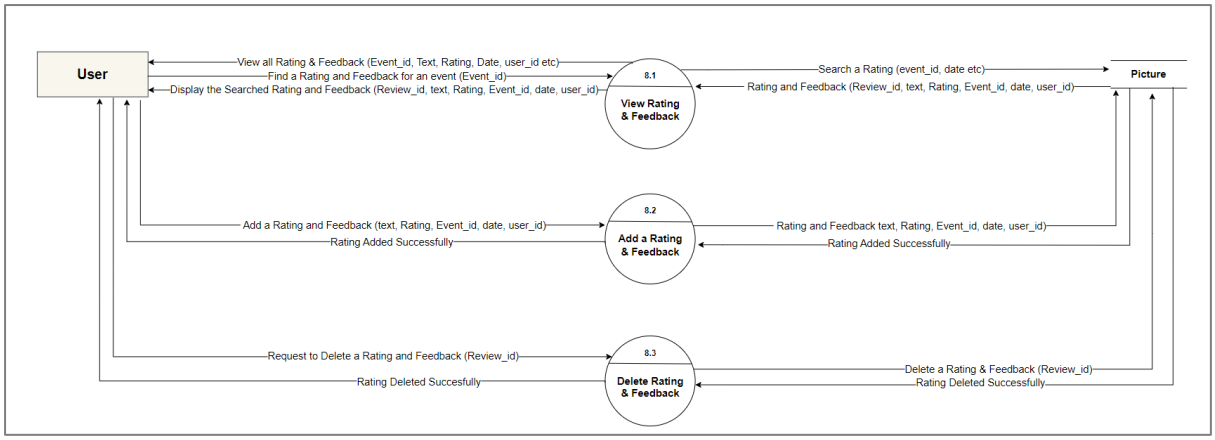
Figures 4.36 DFD Level – 2 for Event Management



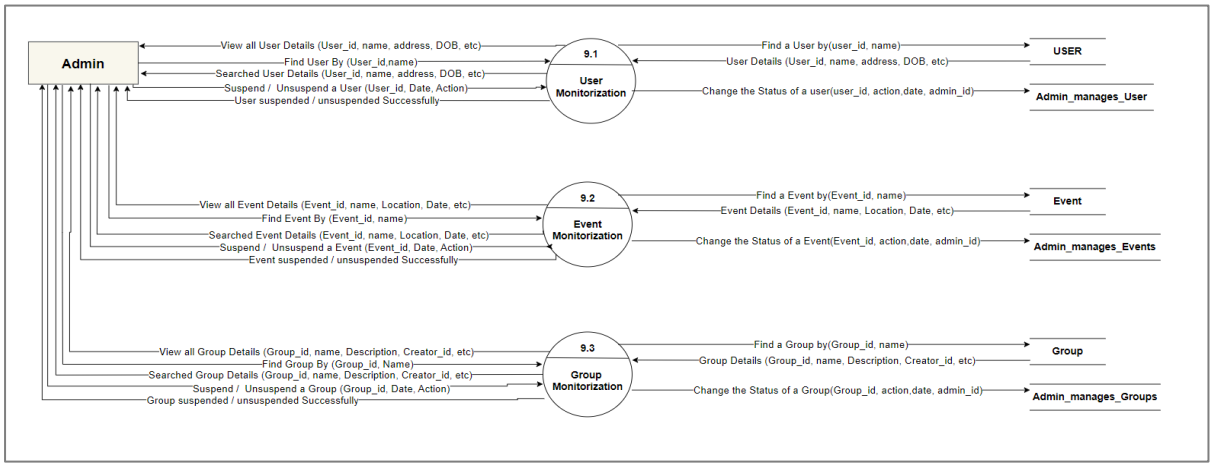
Figures 4.37 DFD Level – 2 for Profile Management



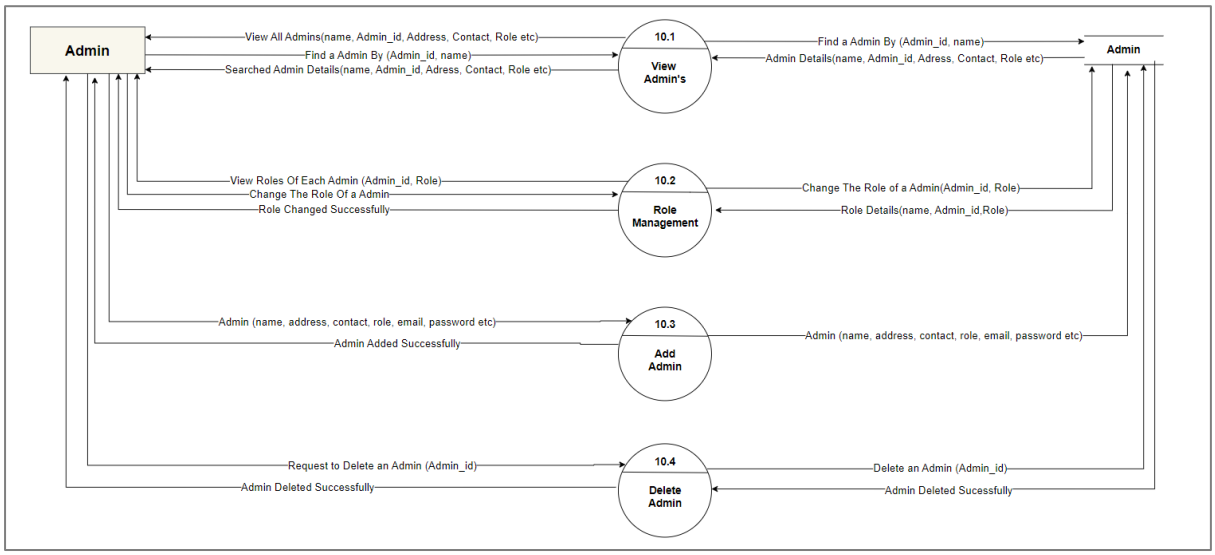
Figures 4.38 DFD Level – 2 for Post Pictures & Comments



Figures 4.39 DFD Level – 2 for Rating and Feedback



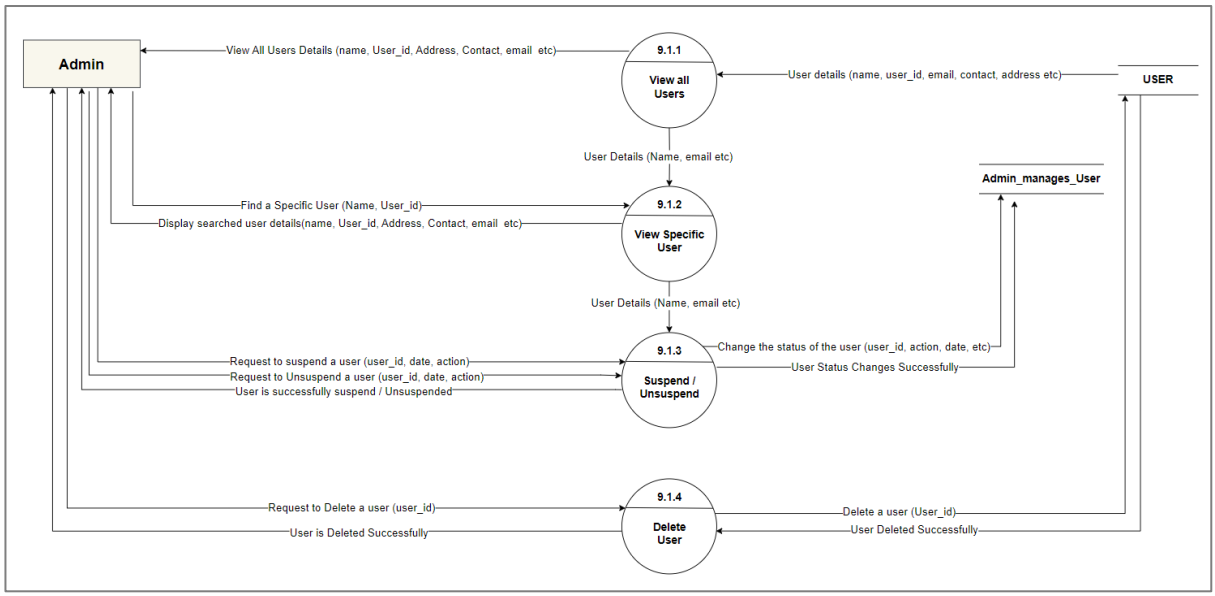
Figures 4.40 DFD Level – 2 for User, Event, and Group Monitorization



Figures 4.41 DFD Level – 2 for Admin Control

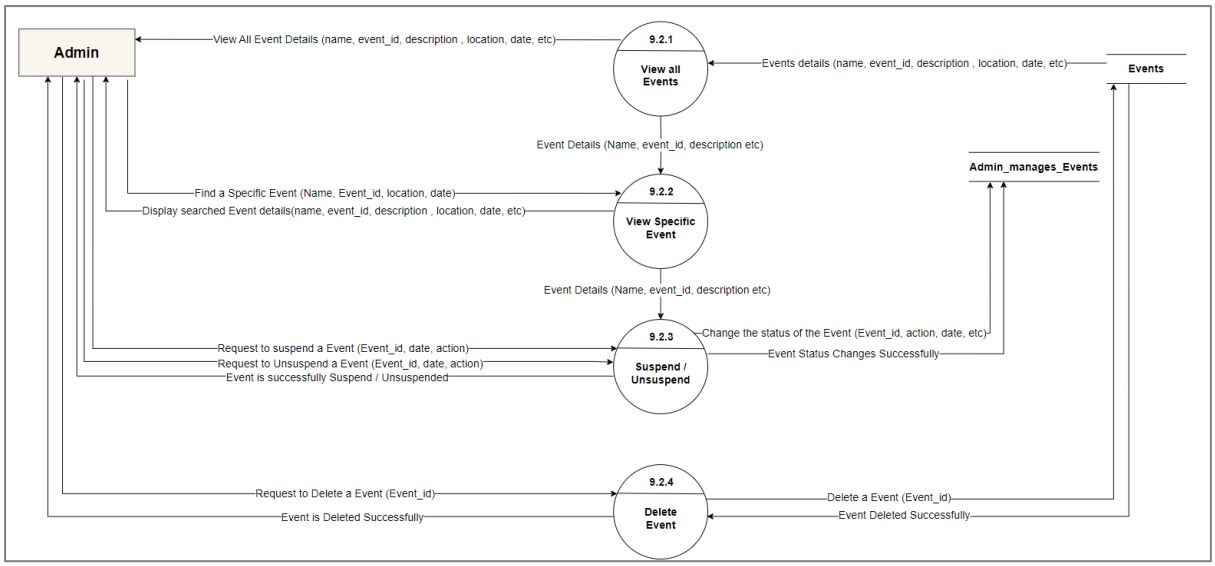
4.4.4 Third Level DFD:

4.4.4.1 Level 3 DFD for User Monitorization:



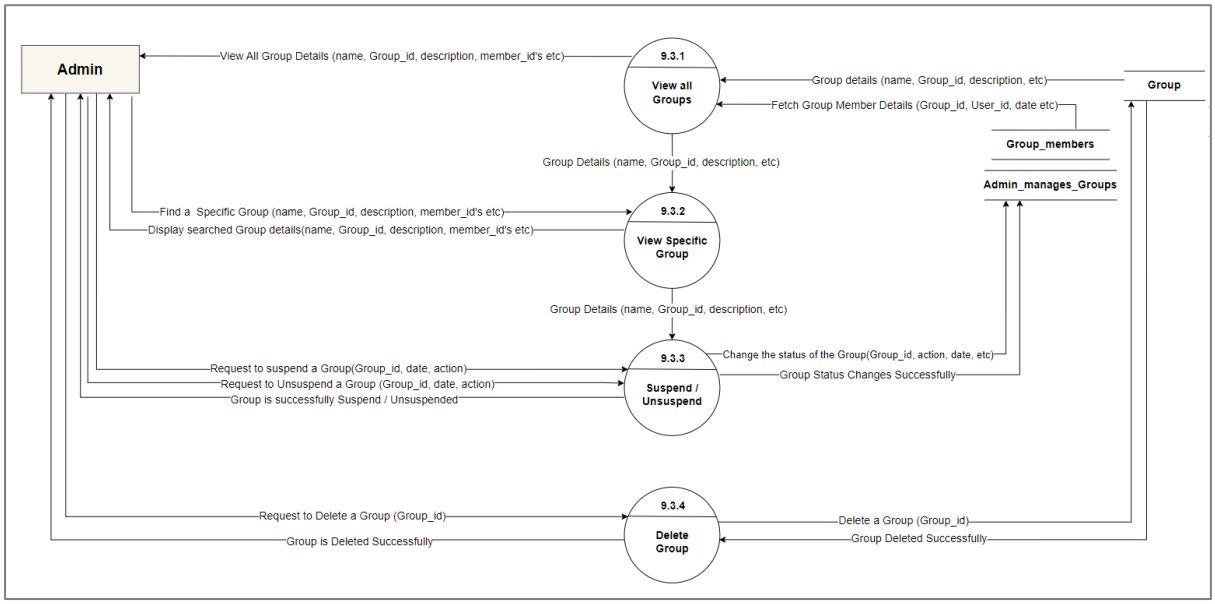
Figures 4.42 DFD Level – 3 for User Monitorization

4.4.4.2 Level 3 DFD for Event Monitorization:



Figures 4.43 DFD Level – 3 for Event Monitorization

4.4.4.3 Level 3 DFD for Group Monitorization:



Figures 4.44 DFD Level – 3 for Group Monitorization

4.5 Use Case Diagram:

Each use case represents a slice of the functionality the system provides. The set of use cases shows the complete functionality of the system at some level of detail. Similarly, each actor represents one kind of object for which the system can perform behaviour. The set of actors represents the complete set of objects that the system can serve. Objects accumulate behaviour from all the systems with which they interact as actors.

Diagram Notations: -

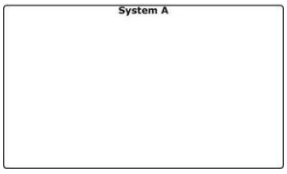
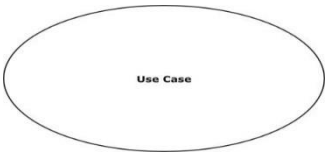
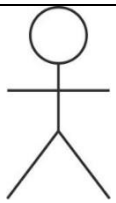
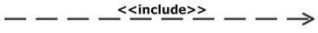
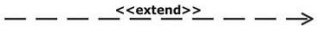
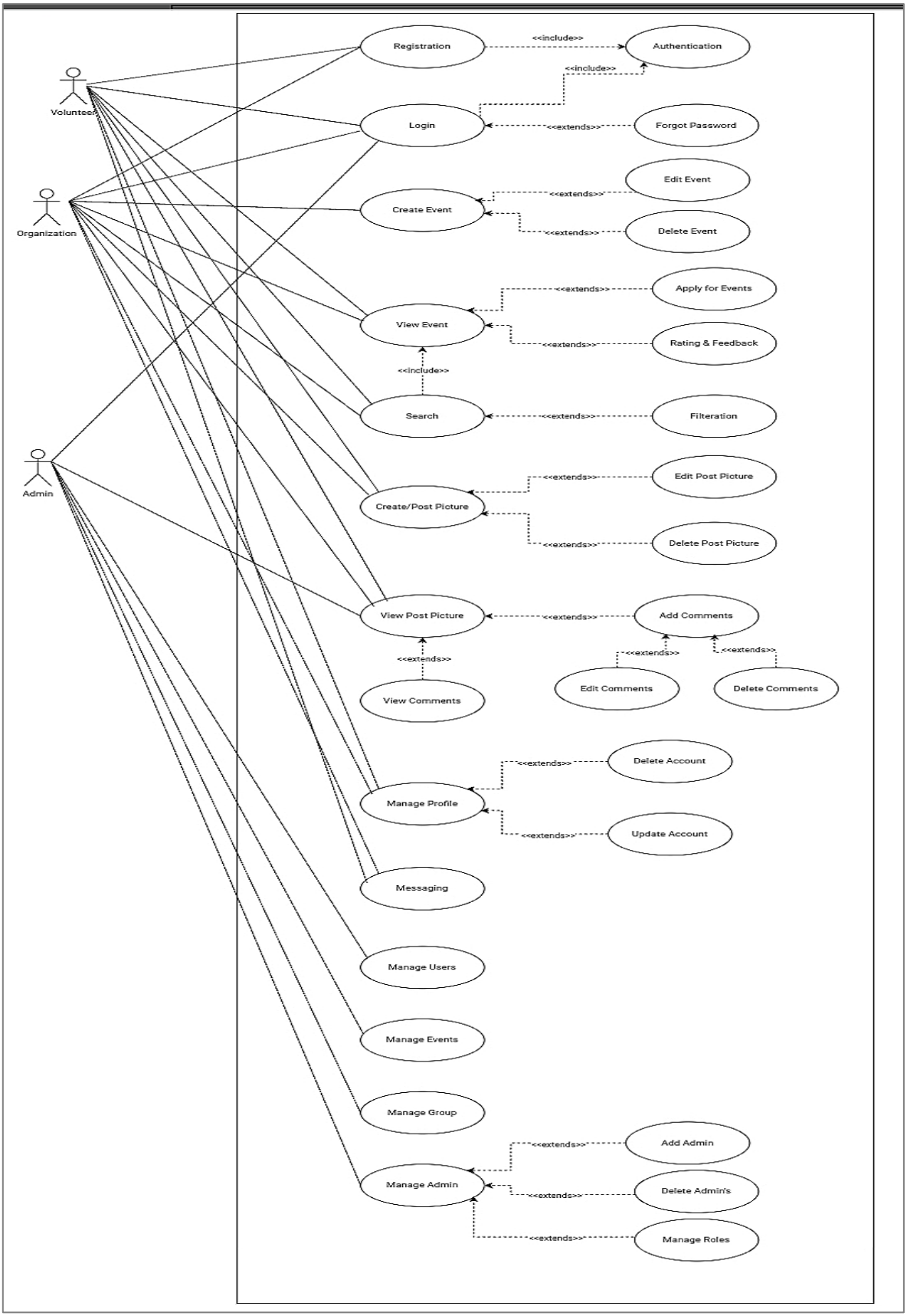
Name	**Symbol**	**Description**
System		The rectangular boundary is the system. Use cases fall inside it, and actors will be placed outside it.
Use Case		An oval shape represents a use case. Use cases represent the functionality of the system, as well as the end-goal of the actor. Use cases should be placed inside the system.
Actor		When an actor interacts with the system, it triggers a use case. Actors should be placed outside the system.
Association		Association between use cases.
Include Relationship		An extension indicates that one use case may include the behaviour of another use case.
Extend Relationship		An inclusion represents one use case using the functionality of another use case.

Table 4.4 Use Case Diagram Notations

4.5.1 Diagram:



Figures 4.45 Use Case Diagram

4.5.2 Use-Case Description:

1. **Registration:**
   - **Description**: The user needs to register by adding details such as email, full name, address, phone number, and password.
   - **Summary**: The user registers for the Volunteer Management System application.
   - **Actor**: User (Volunteer or Organization).
   - **Pre-condition**: The user should not already have an account.
   - **Post-condition**: The user can log in with the registered email and password.
   - **Exception**: An error message is shown if any input is empty, or if the email or phone number does not match the required pattern or is already associated with an existing account.

2. **Login:**
   - **Description**: The user and admin enter their login credentials which are verified by the system. If the verification is successful, the user and admin are redirected to the home page or dashboard.
   - **Summary**: The user and admin enter their credentials to access the system.
   - **Actor**: User (Volunteer or Organization), Admin.
   - **Pre-condition**: The user/Admin must have a registered account .
   - **Post-condition**: The user/Admin is authenticated and is redirected to Homepage.
   - **Exception**: An error message is shown if any input is empty, or if the email or password is incorrect or if the account doesn't exist.

3. **Forgot Password:**
   - **Description**: The user requests to reset their password. An OTP (One-Time Password) is sent to their registered email or phone number for verification. Upon successful verification, the user can reset their password.
   - **Summary**: The user resets their password using OTP verification.

- **Actor**: User (Volunteer or Organization), Admin.
- **Pre-condition**: The user must have a registered account with a valid email.
- **Post-condition**: The user receives an OTP, verifies it, and resets their password.
- **Exception**: An error message is shown if the email is not found in the system, the OTP verification fails, or the new password input does not meet the required criteria.

4. **Create Event:**
   - **Description**: The organization posts details about a new volunteering event, including title, description, date, time, location, and required skills.
   - **Summary**: The organization posts a new event to attract volunteers.
   - **Actor**: Organization.
   - **Pre-condition**: The organization must be registered and logged in.
   - **Post-condition**: The event details are visible to all registered volunteers.
   - **Exception**: An error message is shown if any input is empty or does not meet the required format.

5. **Edit Event:**
   - **Description**: The organization updates details of an existing event, such as changing the date, time, location, or description.
   - **Summary**: The organization modifies the details of a posted event.
   - **Actor**: Organization.
   - **Pre-condition**: The organization must be registered, logged in, and the owner of the event.
   - **Post-condition**: The updated event details are saved and displayed to volunteers.
   - **Exception**: An error message is shown if any input is empty, does not meet the required format, or if the user does not have permission to edit the event.

6. **Delete Event:**
   - **Description**: The organization deletes an existing event from the system.
   - **Summary**: The organization removes an event from the platform.

- **Actor**: Organization.
- **Pre-condition**: The organization must be registered, logged in, and the owner of the event.
- **Post-condition**: The event is removed from the system and is no longer visible to volunteers.
- **Exception**: An error message is shown if the user does not have permission to delete the event or if the deletion process encounters an error.

7. **View Event:**
   - **Description**: The user browses through the list of available volunteering events.
   - **Summary**: The user views details of various events posted on the platform.
   - **Actor**: User (Volunteer).
   - **Pre-condition**: The user must be registered and logged in to see event details.
   - **Post-condition**: The user can see a list of events with details such as title, description, date, time, location, and required skills.
   - **Exception**: An error message is shown if the event data fails to load due to technical issues.

8. **Apply for Event:**
   - **Description**: The volunteer browses through the list of events and applies to participate in an event by clicking the "Apply" button.
   - **Summary**: The volunteer expresses interest and signs up for an event.
   - **Actor**: Volunteer.
   - **Pre-condition**: The volunteer must be registered, logged in, and able to view event details.
   - **Post-condition**: The volunteer's application is recorded, and the organization is notified of the volunteer's interest.
   - **Exception**: An error message is shown if the application process fails due to technical issues or if the event has reached its maximum number of participants or if the volunteer has already applied for the event.

9. **Feedback and Ratings:**
   - **Description**: The volunteer provides feedback and rates an event they participated in.
   - **Summary**: The volunteer submits their evaluation and rating for a completed event.
   - **Actor**: Volunteer.

- **Pre-condition**: The volunteer must be registered, logged in, and have participated in the event.
- **Post-condition**: The feedback and rating are recorded and visible to the organization and other volunteers.
- **Exception**: An error message is shown if the feedback submission fails due to technical issues or if the volunteer has not participated in the event.

10. Search:

- **Description**: The user searches for events based on specific criteria such as keywords, date, location, or category.
- **Summary**: The user locates events that match their search criteria.
- **Actor**: User (Volunteer or Organization).
- **Pre-condition**: The user must be registered and logged in to perform detailed searches.
- **Post-condition**: The system displays a list of events that match the search criteria.
- **Exception**: An error message is shown if no events match the search criteria or if there is a technical issue during the search process.

11. Create/Post Picture:

- **Description**: The user (volunteer or organization) uploads and posts pictures related to volunteering events.
- **Summary**: The user shares images from events to the platform.
- **Actor**: User (Volunteer or Organization).
- **Pre-condition**: The user must be registered, logged in, and the image file must meet the platform's requirements (e.g., file size and format).
- **Post-condition**: The picture is uploaded, stored in the database, and displayed on the event's page.
- **Exception**: An error message is shown if the upload fails due to technical issues, file size limits, or unsupported file formats.

12. Edit Post Picture:

- **Description**: The user edits details of an already posted picture related to a volunteering event.
- **Summary**: The user modifies the description or other details of a previously uploaded picture.
- **Actor**: User (Volunteer or Organization).
- **Pre-condition**: The user must be registered, logged in, and the owner of the picture.
- **Post-condition**: The updated details are saved and displayed on the platform.
- **Exception**: An error message is shown if the user does not have permission to edit the picture.

13. Delete Post Picture:

- **Description**: The user deletes a picture they previously posted related to a volunteering event.
- **Summary**: The user removes an uploaded picture from the platform.

- **Actor**: User (Volunteer or Organization).
- **Pre-condition**: The user must be registered, logged in, and the owner of the picture.
- **Post-condition**: The picture is removed from the platform and is no longer visible to other users.
- **Exception**: An error message is shown if the user does not have permission to delete the picture or if there are technical issues during the deletion process.

14. View Post Picture:

- **Description**: The user views pictures that have been posted related to a volunteering event.
- **Summary**: The user accesses and sees images associated with a specific event.
- **Actor**: User (Volunteer or Organization).
- **Pre-condition**: The user must be registered and logged in to view pictures.
- **Post-condition**: The user can see the uploaded pictures along with their details (such as captions or descriptions).
- **Exception**: None.

15. View Comments:

- **Description**: The user views comments made on a posted picture related to a volunteering event.
- **Summary**: The user accesses and reads comments associated with a specific image that has been uploaded.
- **Actor**: User (Volunteer or Organization).
- **Pre-condition**: The user must be registered and logged in to view comments on the pictures.
- **Post-condition**: The user can see all comments associated with the picture, including the text and commenter's details.
- **Exception**: None.

16. Add Comments:

- **Description**: The user adds a comment to a posted picture related to a volunteering event.
- **Summary**: The user submits a textual comment on a specific image that has been uploaded to the platform.
- **Actor**: User (Volunteer or Organization).
- **Pre-condition**: The user must be registered, logged in, and have access to the picture they want to comment on.
- **Post-condition**: The comment is added to the picture and displayed along with other existing comments.
- **Exception**: if any input field is invalid or empty error message is shown.

17. Delete Comments:

- **Description**: The user deletes a comment they previously made on a posted picture related to a volunteering event.
- **Summary**: The user removes their comment from the picture.

- **Actor**: User (Volunteer or Organization).
- **Pre-condition**: The user must be registered, logged in, and the owner of the comment.
- **Post-condition**: The comment is removed from the picture and is no longer visible to other users.
- **Exception**: An error message is shown if the user does not have permission to delete the comment or if there are technical issues during the deletion process.

18. Manage Profile:

- **Description**: The user updates and manages their profile information, including personal details, contact information, and profile picture.
- **Summary**: The user modifies their profile information to keep it current and accurate.
- **Actor**: User (Volunteer or Organization).
- **Pre-condition**: The user must be registered and logged in to access and update their profile.
- **Post-condition**: The updated profile information is saved and reflected on the user's profile page.
- **Exception**: An error message is shown if the update fails due to technical issues or if the input data does not meet the required criteria (e.g., invalid email format or file size limits for profile picture).

19. Delete Account:

- **Description**: The user permanently deletes their account from the system.
- **Summary**: The user removes their account and all associated data from the platform.
- **Actor**: User (Volunteer or Organization).
- **Pre-condition**: The user must be registered and logged in to initiate the account deletion process.
- **Post-condition**: The user's account and all associated data (including posts, comments, and application history) are permanently deleted from the system.
- **Exception**: An error message is shown if the deletion process fails due to technical issues or if the user does not have permission to delete the account. Additionally, confirmation is required to prevent accidental deletions.

20. Messaging:

- **Description**: The user sends and receives messages to and from other users on the platform.
- **Summary**: The user communicates with other users through personal or group messages.
- **Actor**: User (Volunteer or Organization).
- **Pre-condition**: The user must be registered, logged in, and have an active connection to the messaging service.
- **Post-condition**: Messages are sent to and received from other users and are stored in the user's message history.

- **Exception**: if any input field is invalid or empty an error message is shown.

21. Manage Event Applicant:

- **Description**: The event organizer manages applicants who have applied to participate in their event, including reviewing applications, approving or rejecting them.
- **Summary**: The organizer processes and manages volunteer applications for an event.
- **Actor**: Event Organizer.
- **Pre-condition**: The organizer must be registered, logged in, and have access to the event's applicant list.
- **Post-condition**: The organizer's decisions (approve or reject) are recorded, and applicants are notified of their application status.
- **Exception**: An error message is shown if there are issues with the application management process, such as technical failures, or if the organizer does not have the proper permissions to manage the applicants.

22. Manage Users:

- **Description**: The admin manages user accounts by suspending or unsuspending users, which involves restricting or restoring their access to the platform.
- **Summary**: The admin can temporarily disable or enable user accounts as needed.
- **Actor**: Admin.
- **Pre-condition**: The admin must be logged in with appropriate permissions to manage user accounts.
- **Post-condition**: The targeted user's account status is updated to suspended or active, and the user is notified of the change in their access rights.
- **Exception**: An error message is shown if the suspension or unsuspension fails due to technical issues or if the admin lacks the necessary permissions to perform the action.

23. Manage Events:

- **Description**: The admin manages events by suspending or unsuspending them, which involves temporarily disabling or restoring the event's visibility and participation.
- **Summary**: The admin can control the status of events by suspending them (making them inactive or hidden) or unsuspending them (making them active or visible).
- **Actor**: Admin.
- **Pre-condition**: The admin must be logged in with the necessary permissions to manage events.
- **Post-condition**: The event's status is updated to suspended or active, and users are informed of the change in the event's availability.
- **Exception**: None.

24. Manage Groups:

- **Description**: The admin manages user groups by suspending or unsuspending them, which involves temporarily disabling or restoring the group's visibility and functionality on the platform.
- **Summary**: The admin can control the status of user groups by suspending them (making them inactive or hidden) or unsuspending them (making them active or visible).
- **Actor**: Admin.
- **Pre-condition**: The admin must be logged in with the appropriate permissions to manage user groups.
- **Post-condition**: The group's status is updated to suspended or active, and group members are notified of the change in the group's status.
- **Exception**: None.

25. Add Admin:

- **Description**: The existing admin adds a new admin to the system, granting them administrative privileges to manage various aspects of the platform.
- **Summary**: The current admin creates a new admin account with the required permissions.
- **Actor**: Existing Admin (Super Admin).
- **Pre-condition**: The existing admin must be logged in with the necessary permissions to add new admin accounts.
- **Post-condition**: A new admin account is created and the new admin is notified of their access and responsibilities.
- **Exception**: An error message is shown if the account creation fails due to technical issues or if the existing admin lacks the necessary permissions to perform this action.

26. Delete Admin:

- **Description**: The existing admin removes an admin account from the system, revoking their administrative privileges and access.
- **Summary**: The current admin deletes an admin account, ensuring the individual no longer has administrative control.
- **Actor**: Existing Admin (Super Admin).
- **Pre-condition**: The existing admin must be logged in with the necessary permissions to delete admin accounts.
- **Post-condition**: The targeted admin account is deleted, and the former admin is notified of the revocation of their access and privileges.
- **Exception**: An error message is shown if the account deletion fails due to technical issues or if the existing admin lacks the necessary permissions to perform this action. Additionally, the system should ensure at least one admin account remains active to prevent locking out administrative access.

27. Role Management:

- **Description**: The admin manages user roles within the system, including assigning, modifying, or removing roles for different users to control their access and permissions.
- **Summary**: The admin controls the roles of users, determining what actions they can perform on the platform.

- **Actor**: Admin (Super Admin).
- **Pre-condition**: The admin must be logged in with the necessary permissions to manage user roles.
- **Post-condition**: The user's role is updated, and they are notified of any changes to their permissions and access rights.
- **Exception**: None.

4.6 Scenario:

1. Registration:

i. The user navigates to the registration page and inputs the required details (name, email, phone number, password, date of Birth, username etc).
ii. The system checks if the email or phone number is already associated with an existing account.
iii. The system sends an email containing a verification link to the entered email address.
iv. The user clicks on the verification link in the email.
v. The system verifies the link and confirms the user's email address.
vi. If verification is successful, the user's details are stored in the database, and a "registration successful" message is displayed.
vii. If any error occurs during registration, an "registration unsuccessful" message is displayed.
viii. The user is then redirected to the login page.

2. Login:

i. The user navigates to the login page and inputs their registered email and password.
ii. The system checks if the email and password match an existing account.
iii. If the credentials are correct, the user is granted access to the system and redirected to the homepage.
iv. If the credentials are incorrect, an error message is displayed, and the user is prompted to try again.

3. Forgot Password:

i. The user issues a request to reset their password
ii. The user enters an email.
iii. The system checks if the email is registered with an user or not.
iv. If the email is a valid registered email:
    a) The system sends a email containing the OTP to the users email address.
    b) The user enters the OTP on the system.
    c) If the OTP is correct, the user is prompted to enter a new password.
    d) The system updates the user's password in the database.
    e) A "Password rese Successful" message is displayed, and the user is redirected to the login page.
    f) If the OTP is incorrect or any error occurs, an appropriate error message is displayed.
v. Else, "OTP verification Failed ! Enter a valid email !" message is displayed.

4. **Add/Create Event:**
   i. The organizer navigates to the "Add Event" page.
   ii. The organizer inputs event details such as title, date, time, location, description, and any other relevant information.
   iii. The system validates the input details to ensure all required fields are filled and the data is in the correct format.
   iv. If the input is valid, the system stores the event details in the database.
   v. A "event added successfully" message is displayed to the organizer.
   vi. If any error occurs during the process, an "event addition unsuccessful" message is displayed, and the organizer is prompted to correct the errors.

5. **Edit Event Details:**
   i. The organizer navigates to their event management page and selects an event to edit.
   ii. The organizer makes changes to the event details (e.g., date, time, location, description).
   iii. The system validates the new event details.
   iv. If the details are valid, the system updates the event information in the database.
   v. A "event updated successfully" message is displayed to the organizer.
   vi. If any error occurs, an "event update unsuccessful" message is displayed.

6. **Delete Event:**
   i. The organizer navigates to their event management page and selects an event to delete.
   ii. The system prompts the organizer to confirm the deletion.
   iii. If the organizer confirms, the system removes the event from the database.
   iv. A "event deleted successfully" message is displayed.
   v. If any error occurs, an "event deletion unsuccessful" message is displayed.

7. **View Event:**
   i. The user navigates to the events page.
   ii. The system retrieves the list of all active events from the database.
   iii. The events are displayed to the user with details such as date, time, location, and description.
   iv. The user can click on an event to view more details or to apply for the event.

8. **Search Event:**
   i. The user navigates to the search page and inputs keywords or filters for events.
   ii. The system retrieves events from the database that match the search criteria.
   iii. The matching events are displayed to the user.
   iv. The user can click on an event to view more details or to apply for the event.

9. **Apply for Event:**
   i. The user views the details of an event and clicks the "Apply" button.

ii. The system checks if the user is logged in. If not, the user is prompted to log in.
iii. If the user is logged in, the system records the user's application for the event in the database.
iv. A "application successful" message is displayed to the user.
v. If any error occurs, an "application unsuccessful" message is displayed.

10. Feedback and Rating For Events:

i. After participating in an event, the user navigates to the feedback page for that event.
ii. The user inputs their feedback and rating for the event.
iii. The system validates the feedback and rating.
iv. If the input is valid, the system stores the feedback and rating in the database.
v. A "feedback submitted successfully" message is displayed to the user.
vi. If any error occurs, an "feedback submission unsuccessful" message is displayed.

11. Post Picture:

i. The user navigates to the event's page and clicks the "Post Picture" button.
ii. The user uploads a picture and adds a description.
iii. The system validates the picture and description.
iv. If the input is valid, the system stores the picture and description in the database.
v. A "picture posted successfully" message is displayed to the user.
vi. If any error occurs, an "picture posting unsuccessful" message is displayed.

12. Edit Post Picture:

i. The user navigates to their posted pictures page and selects a picture to edit.
ii. The user makes changes to the picture's description.
iii. The system validates the new description.
iv. If the input is valid, the system updates the picture's information in the database.
v. A "picture updated successfully" message is displayed to the user.
vi. If any error occurs, an "picture update unsuccessful" message is displayed.

13. Delete Post Picture:

i. The user navigates to their posted pictures page and selects a picture to delete.
ii. The system prompts the user to confirm the deletion.
iii. If the user confirms, the system removes the picture from the database.
iv. A "picture deleted successfully" message is displayed.
v. If any error occurs, an "picture deletion unsuccessful" message is displayed.

14. View Post Picture:

i. The user navigates to the event's page and views the posted pictures.
ii. The system retrieves all pictures related to the event from the database.
iii. The pictures are displayed to the user with their descriptions.
iv. The user can click on a picture to view it in full size and read the comments.

15. View Comments on Post Picture:

i. The user clicks on a picture to view it in full size.
ii. The system retrieves all comments related to the picture from the database.
iii. The retrieved comments are then displayed to the user, if no comment are found then “No Comments Found!!” is displayed.

16. Add Comments:

i. The user views a picture and inputs a comment in the comment box.
ii. The system validates the comment.
iii. If the comment is valid, the system stores the comment in the database.
iv. A "comment added successfully" message is displayed to the user.
v. If any error occurs, an "comment addition unsuccessful" message is displayed.

17. Delete Comments:

i. The user navigates to their comment on a picture and selects it to delete.
ii. The system prompts the user to confirm the deletion.
iii. If the user confirms, the system removes the comment from the database.
iv. A "comment deleted successfully" message is displayed.
v. If any error occurs, an "comment deletion unsuccessful" message is displayed.

18. Manage Profile:

i. The user navigates to their profile page and clicks the "Edit Profile" button.
ii. The user updates their profile details (e.g., name, email, phone number).
iii. The system validates the new profile details.
iv. If the details are valid, the system updates the user's information in the database.
v. A "profile updated successfully" message is displayed to the user.
vi. If any error occurs, an "profile update unsuccessful" message is displayed.

19. Delete Account:

i. The user navigates to their account settings page and clicks the "Delete Account" button.
ii. The system prompts the user to confirm the account deletion.
iii. If the user confirms, the system deletes the user's account from the database.
iv. A "account deleted successfully" message is displayed.
v. If any error occurs, an "account deletion unsuccessful" message is displayed.

20. Messaging:

i. The user navigates to the messaging page and selects a contact or group to send a message to.
ii. The user types a message and clicks the "Send" button.
iii. The system sends the message to the selected contact or group.
iv. The message is stored in the database and displayed in the chat window.
v. If any error occurs, an "message sending unsuccessful" message is displayed.

21. Manage Event Applicant:

i. The organizer navigates to their event management page and selects an event.
ii. The organizer views the list of applicants for the event.
iii. The organizer reviews the applications and makes decisions (approve or reject).
iv. The system updates the status of each applicant in the database.
v. The applicants are notified of their application status (approved or rejected).
vi. If any error occurs, an appropriate error message is displayed.

22. Manage Users(Suspend/Unsuspend):

i. The admin navigates to the user management page and selects a user to suspend or unsuspend.
ii. The admin changes the status of the user to suspended or active.
iii. The system updates the user's status in the database.
iv. The user is notified of the change in their account status.
v. If any error occurs, an appropriate error message is displayed.

23. Manage Events(Suspend/Unsuspend):

i. The admin navigates to the Event management page and selects a Event to suspend or unsuspend.
ii. The admin changes the status of the Event to suspended or active.
iii. The system updates the Event's status in the database.
iv. Users are notified of the change in the Event's availability.

24. Manage Groups(Suspend/Unsuspend):

i. The admin navigates to the Group management page and selects a Group to suspend or unsuspend.
ii. The admin changes the status of the Group to suspended or active.
iii. The system updates the Group's status in the database.
iv. Users are notified of the change in the Group's availability

4.7 Sequence Diagram:

A sequence diagram in a Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams typically are associated with use case realizations in the Logical View of the system under development.

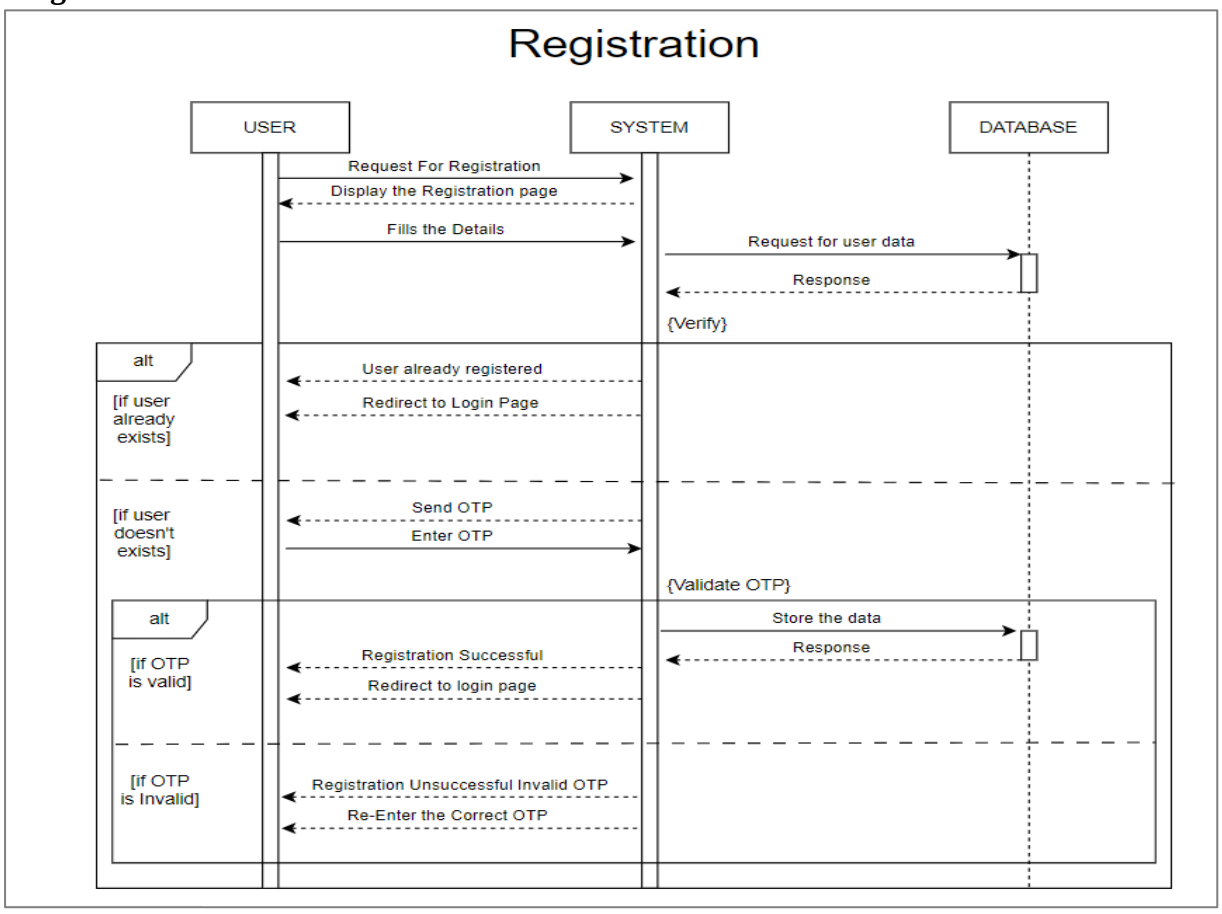
Diagram Notations:

Name	Symbol	Description

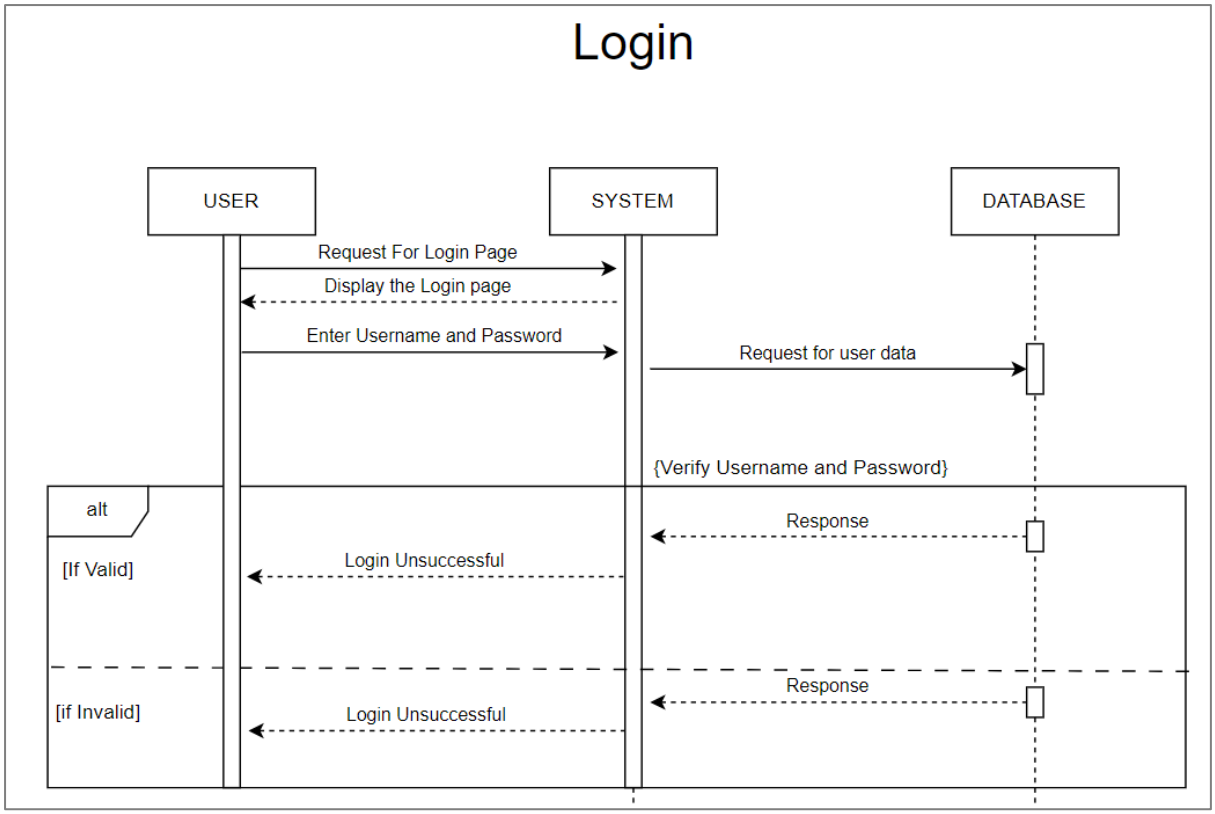
Synchronous Message		An instantaneous communication between objects that conveys information, with the expectation that an action will be initiated as a result.
Activation Box		Represents an active object during an interaction between two objects. The length of the bar represents the duration of an object's activeness.
Object		An object that is created, performs actions, and/or is destroyed during the lifeline
Alternative Fragment		Two or more message sequences exist, and a choice must be made between the two of them.

Table 4.5 Sequence Diagram Notations Table

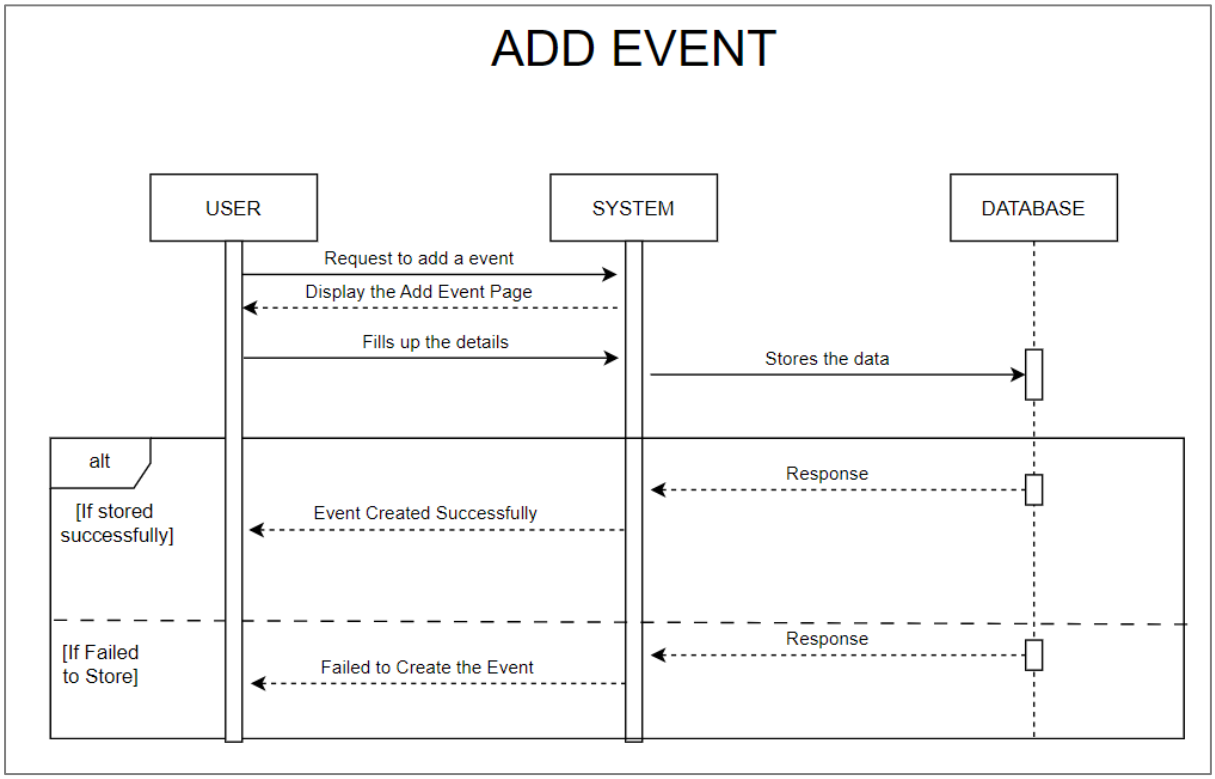
Diagram:



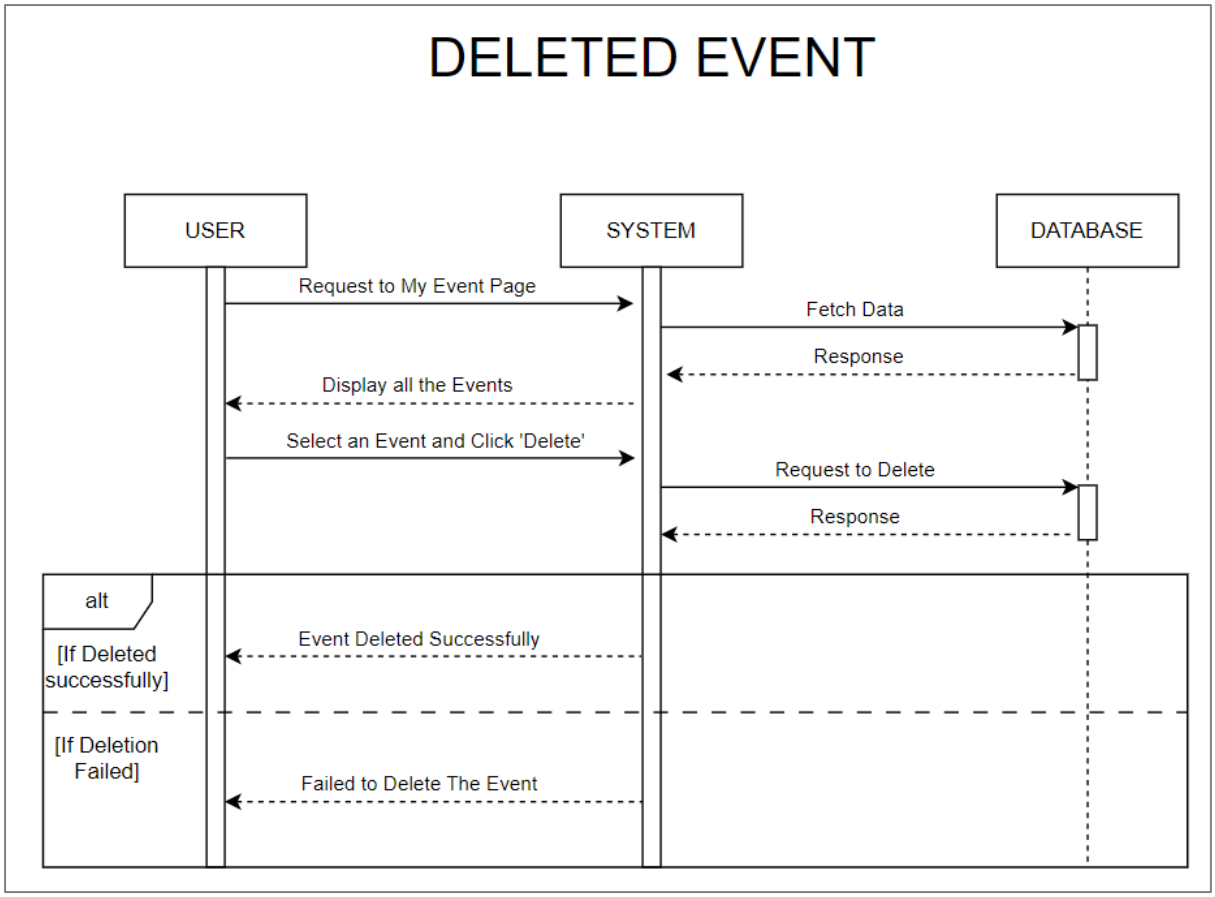
Figures 4.46 Sequence Diagram – Register



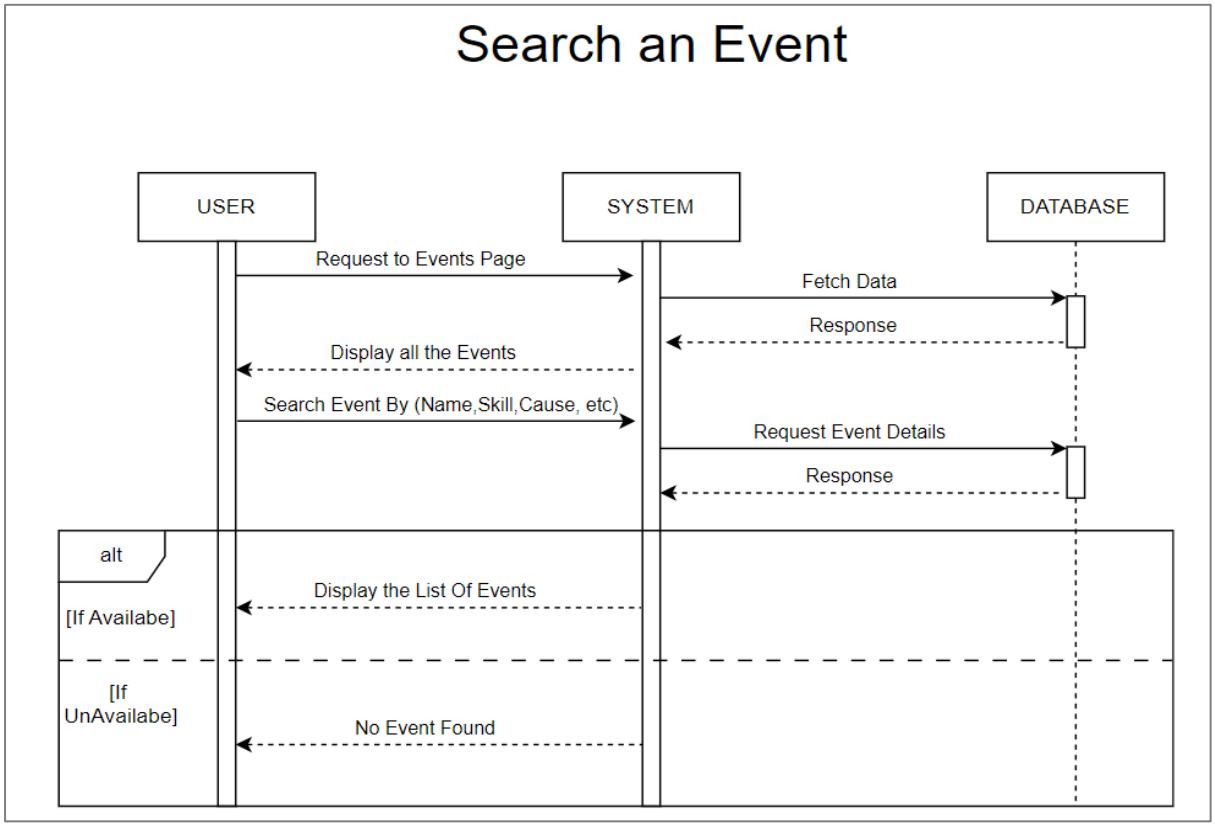
Figures 4.47 Sequence Diagram – Login



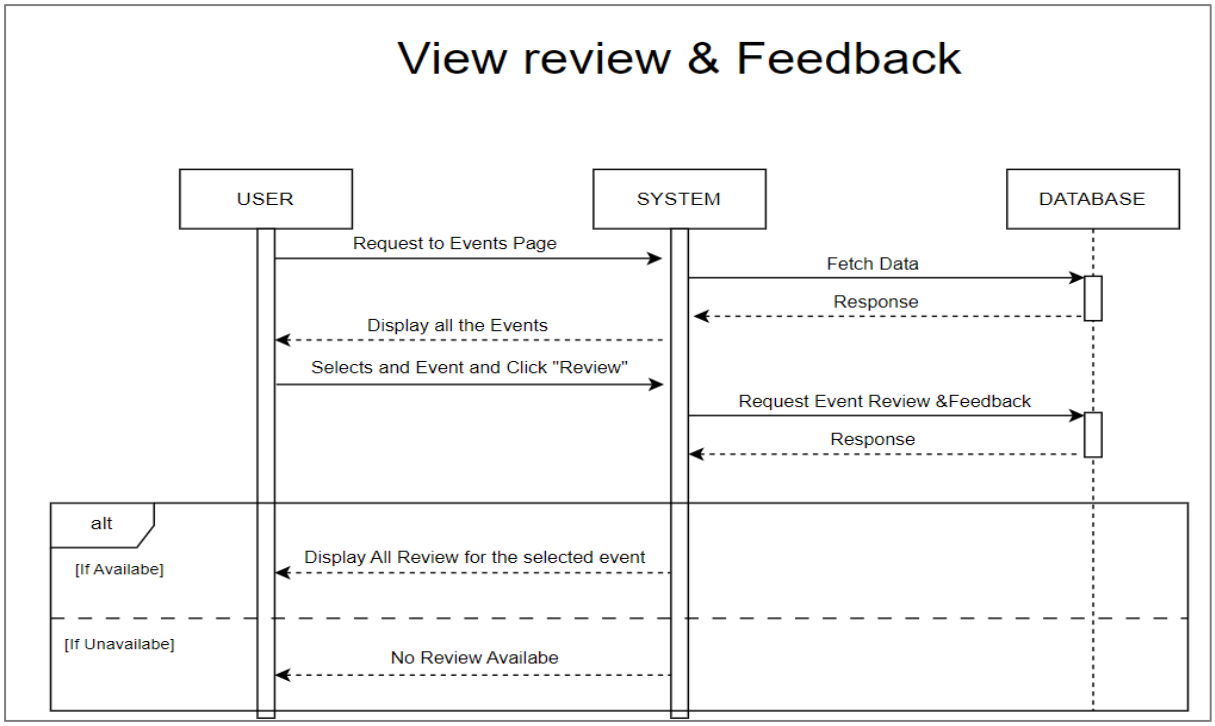
Figures 4.48 Sequence Diagram – Add Event



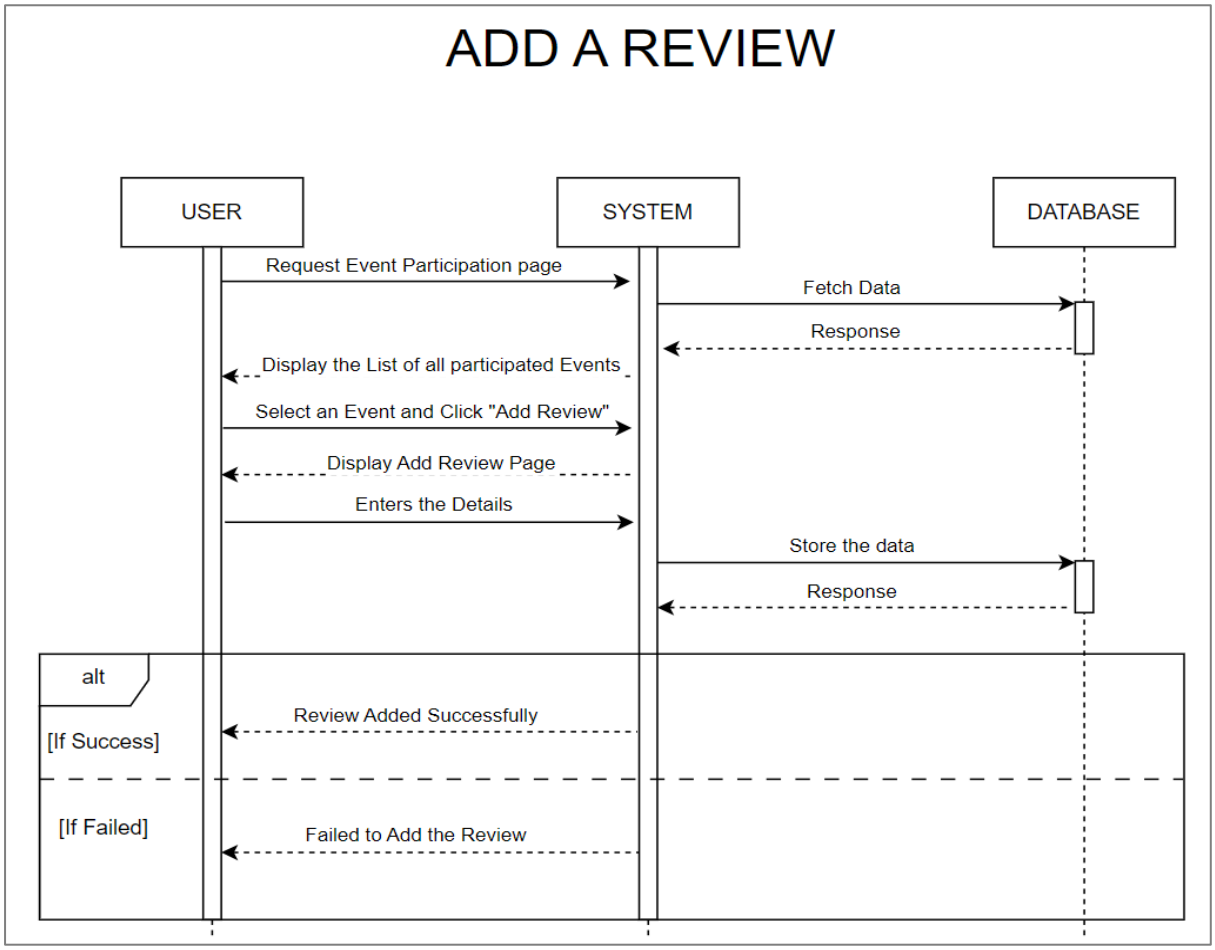
Figures 4.49 Sequence Diagram – Delete Event



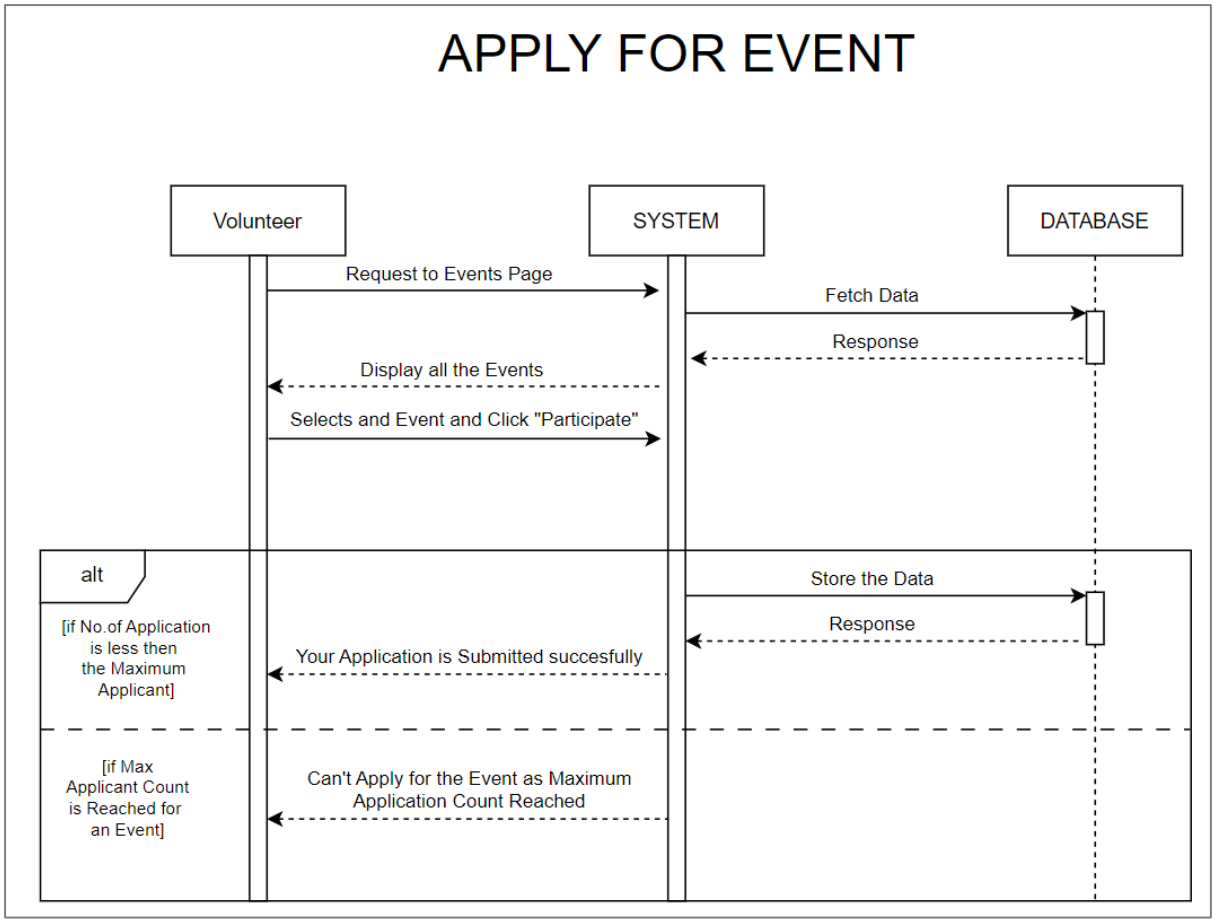
Figures 4.50 Sequence Diagram – Search an Event



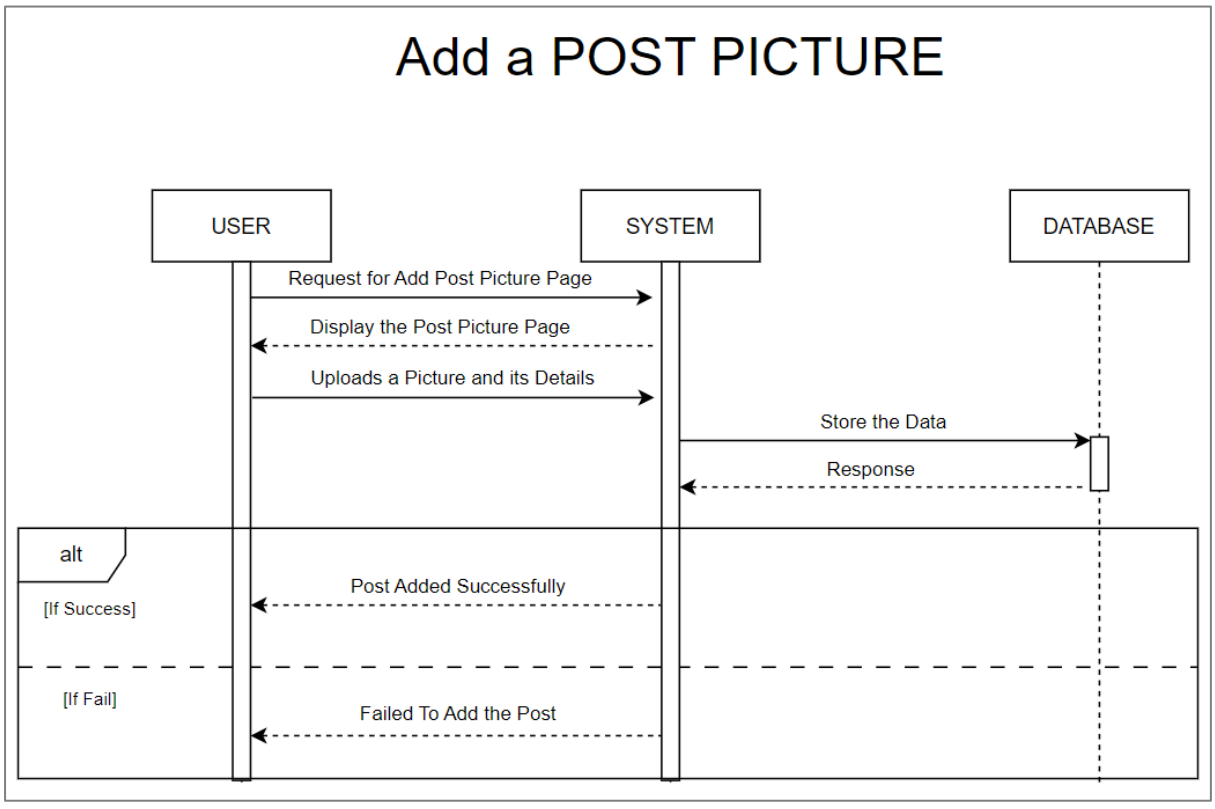
Figures 4.51 Sequence Diagram – View Review & Feedback



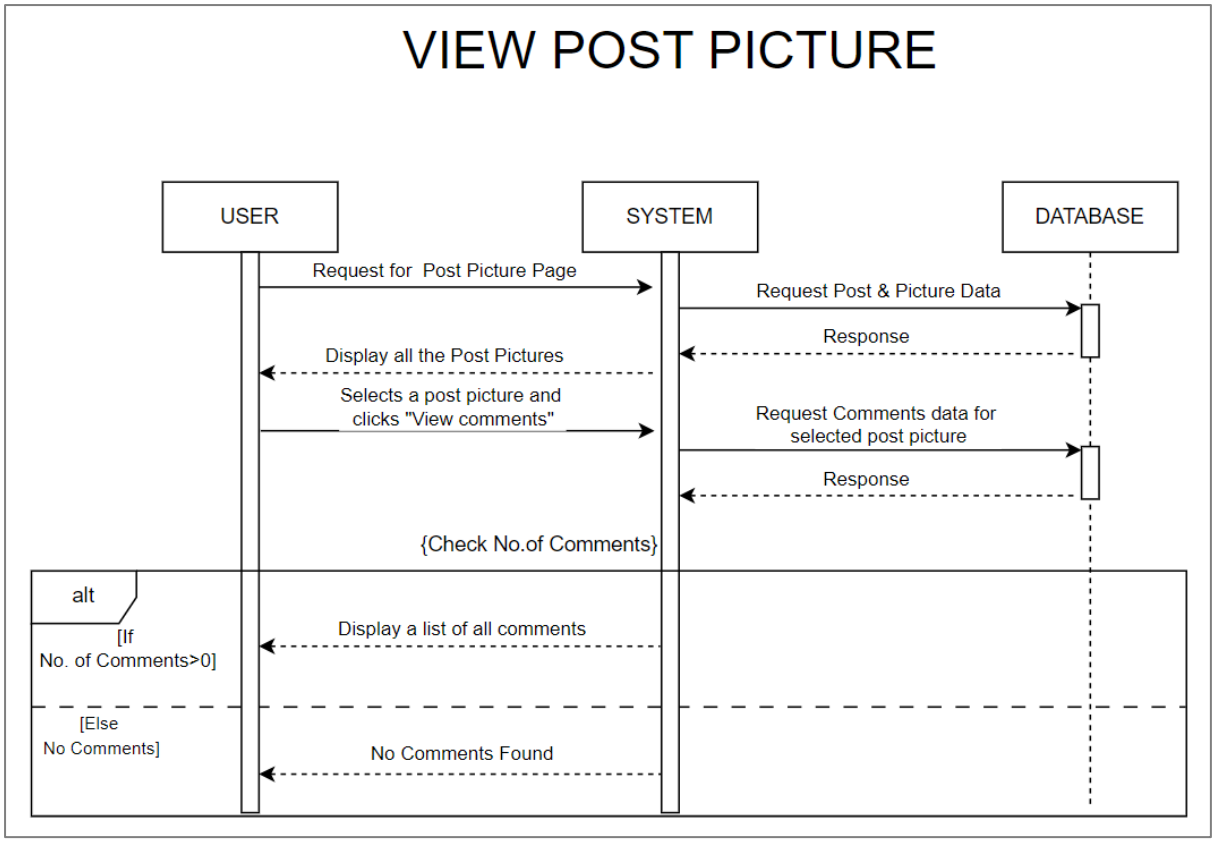
Figures 4.52 Sequence Diagram – Add a Review



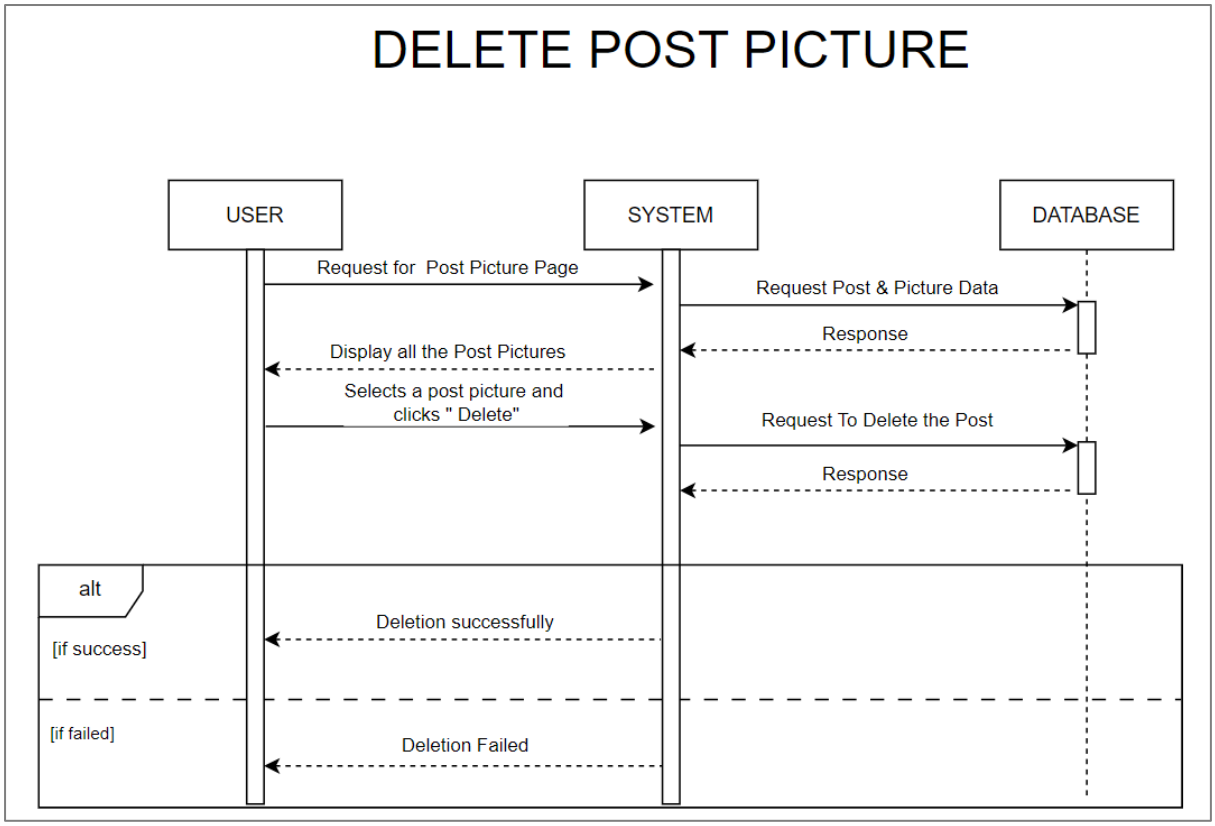
Figures 4.53 Sequence Diagram – Apply for Event



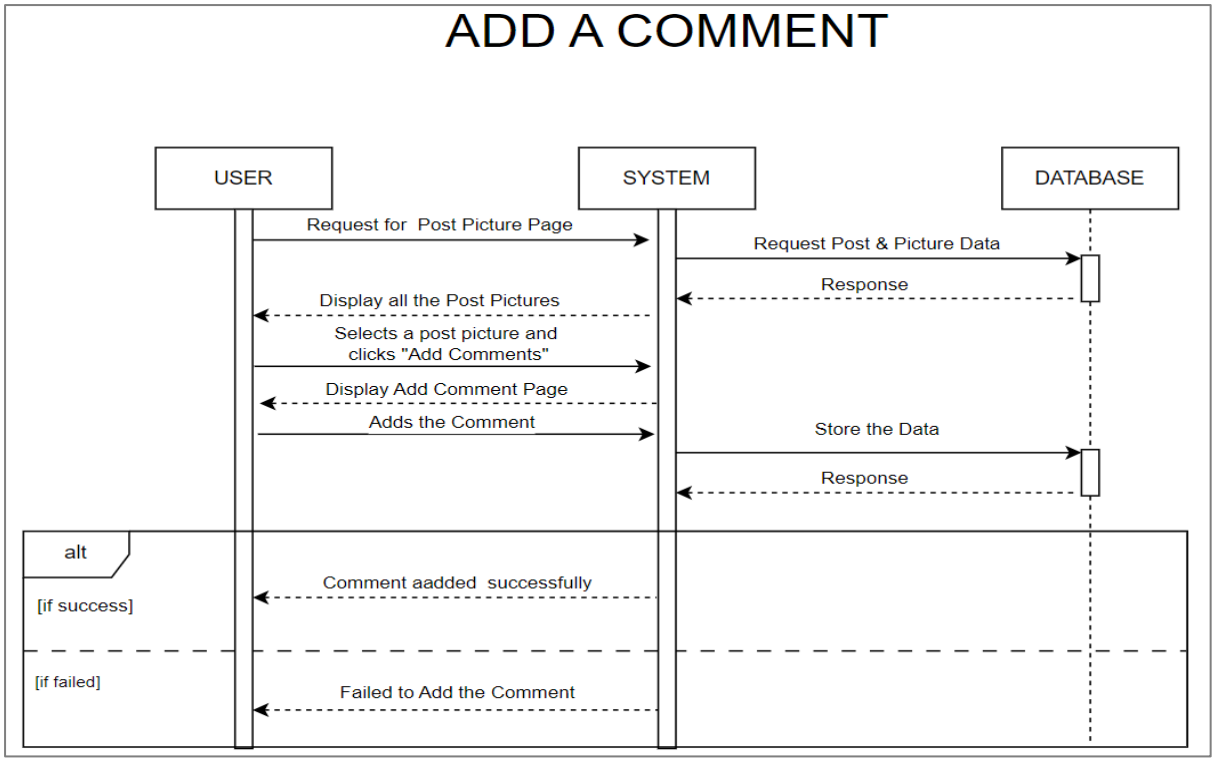
Figures 4.54 Sequence Diagram – Add a Post Picture



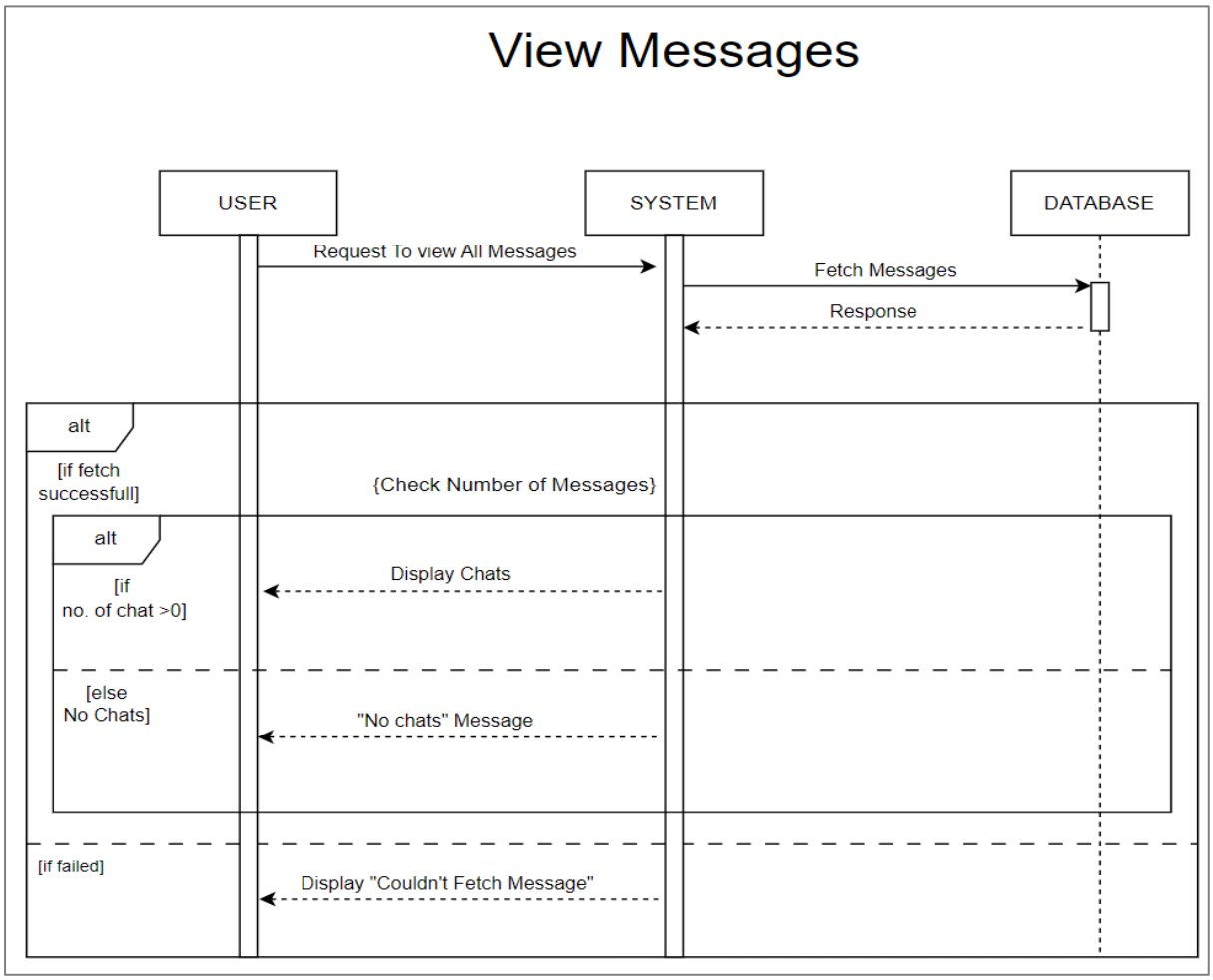
Figures 4.55 Sequence Diagram – View Post Picture



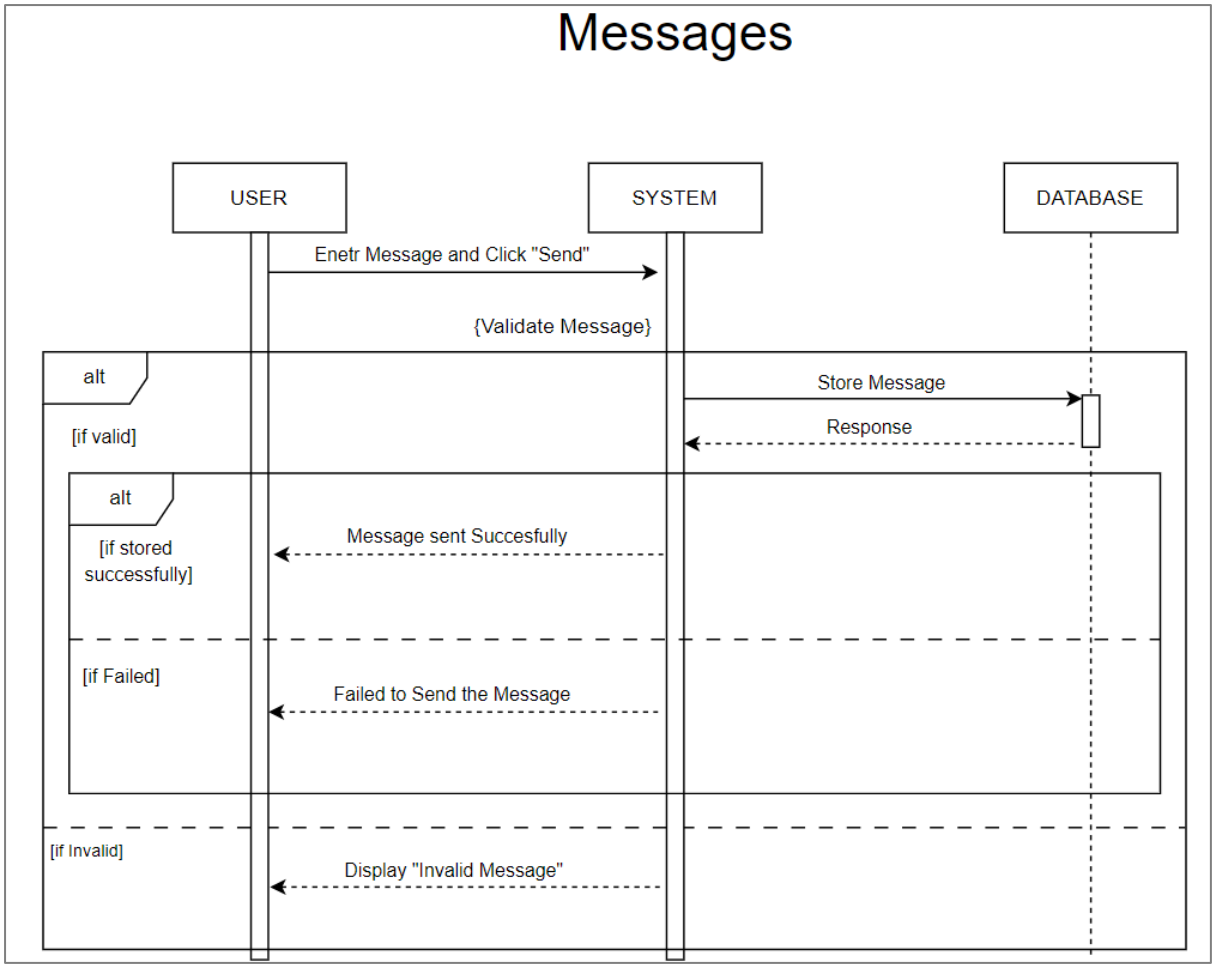
Figures 4.56 Sequence Diagram – Delete Post Picture



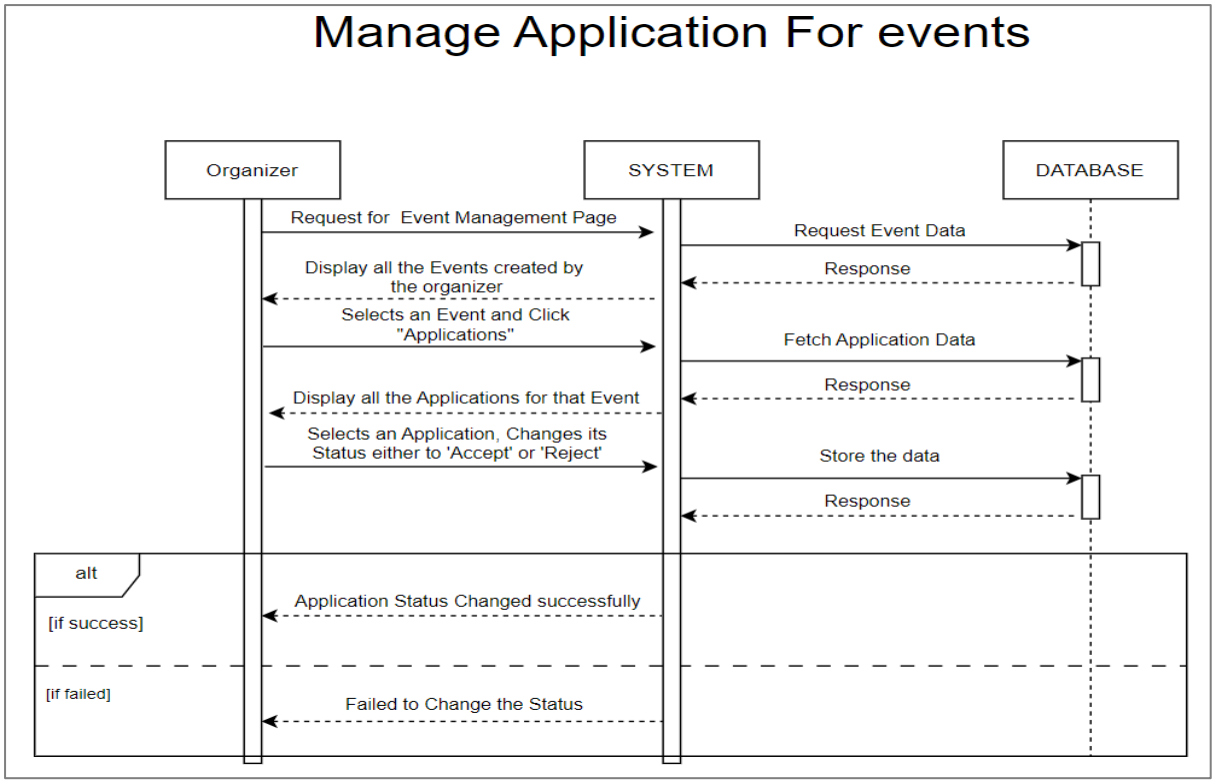
Figures 4.57 Sequence Diagram – Add a Comment



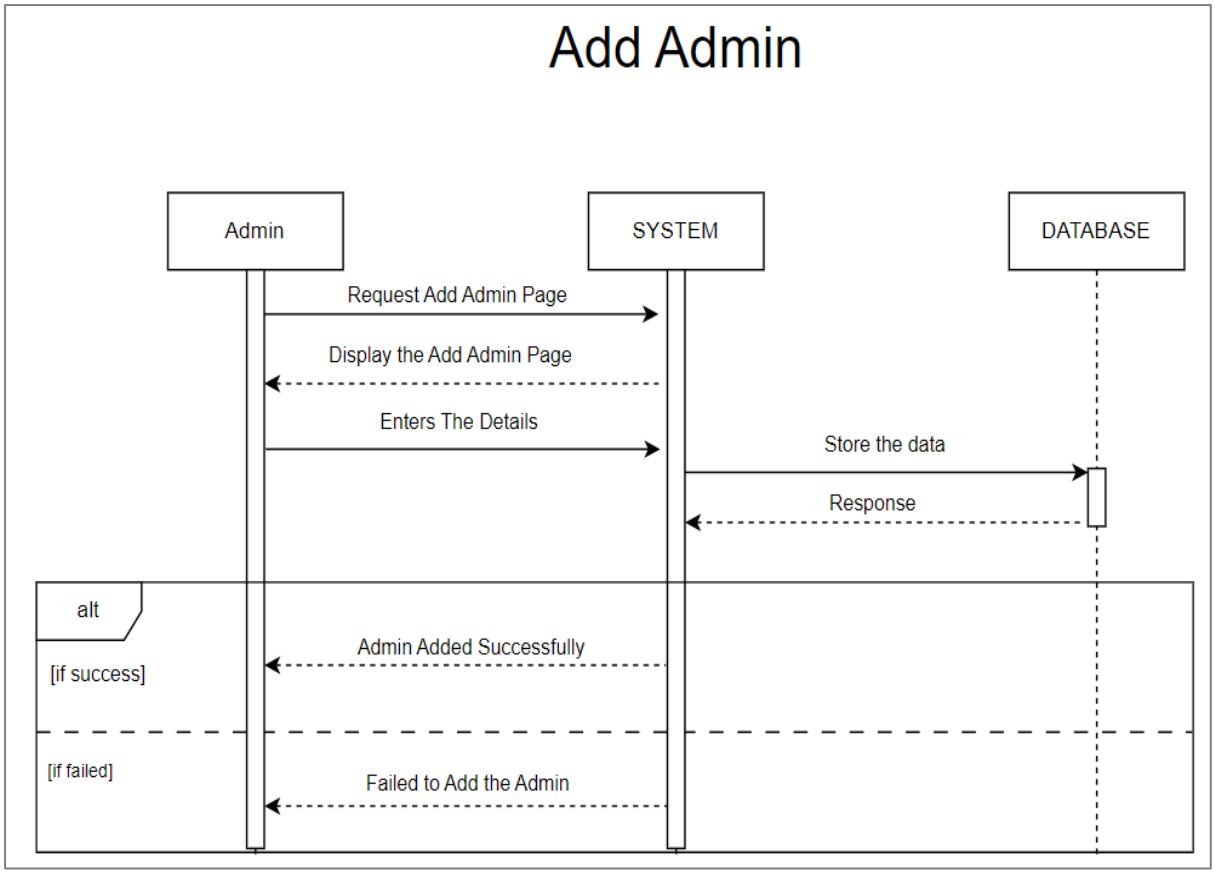
Figures 4.58 Sequence Diagram – View Messages



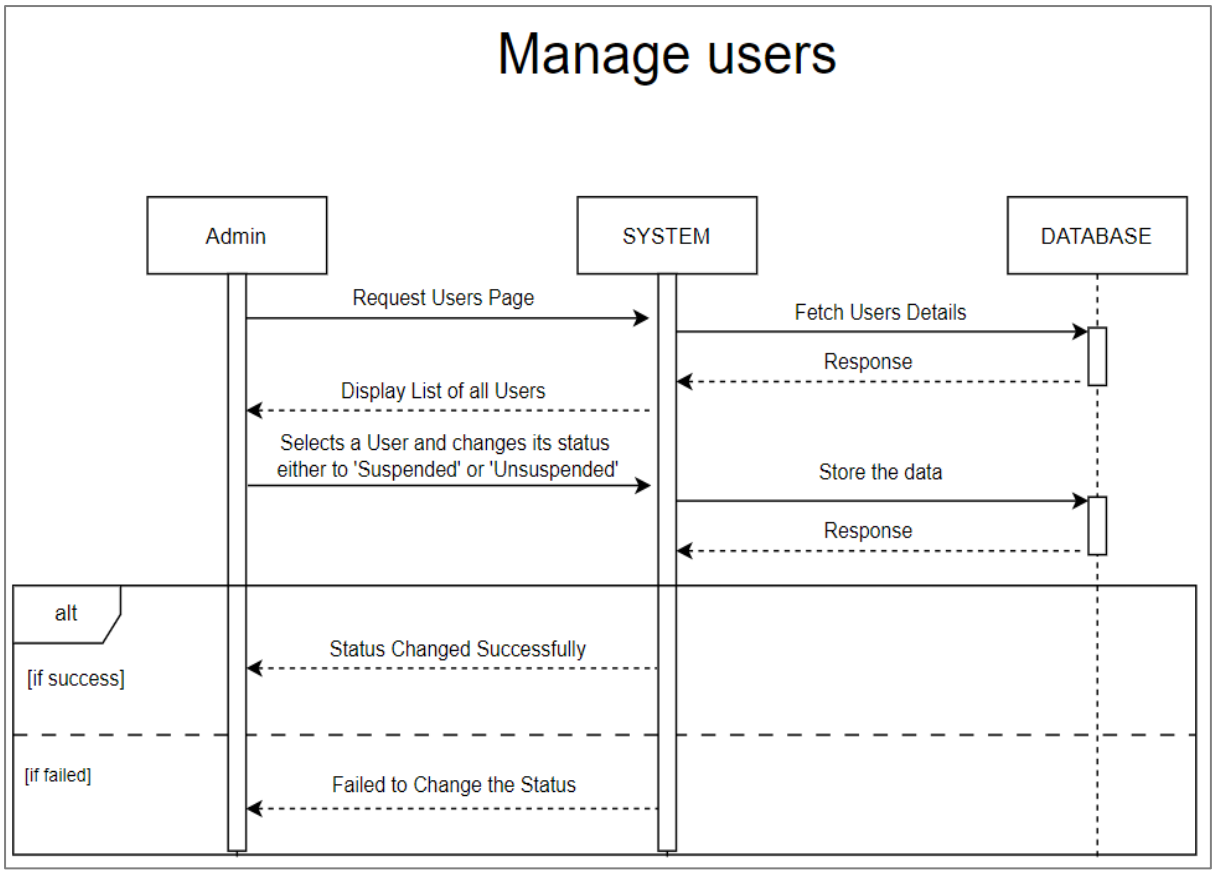
Figures 4.59 Sequence Diagram – Messaging



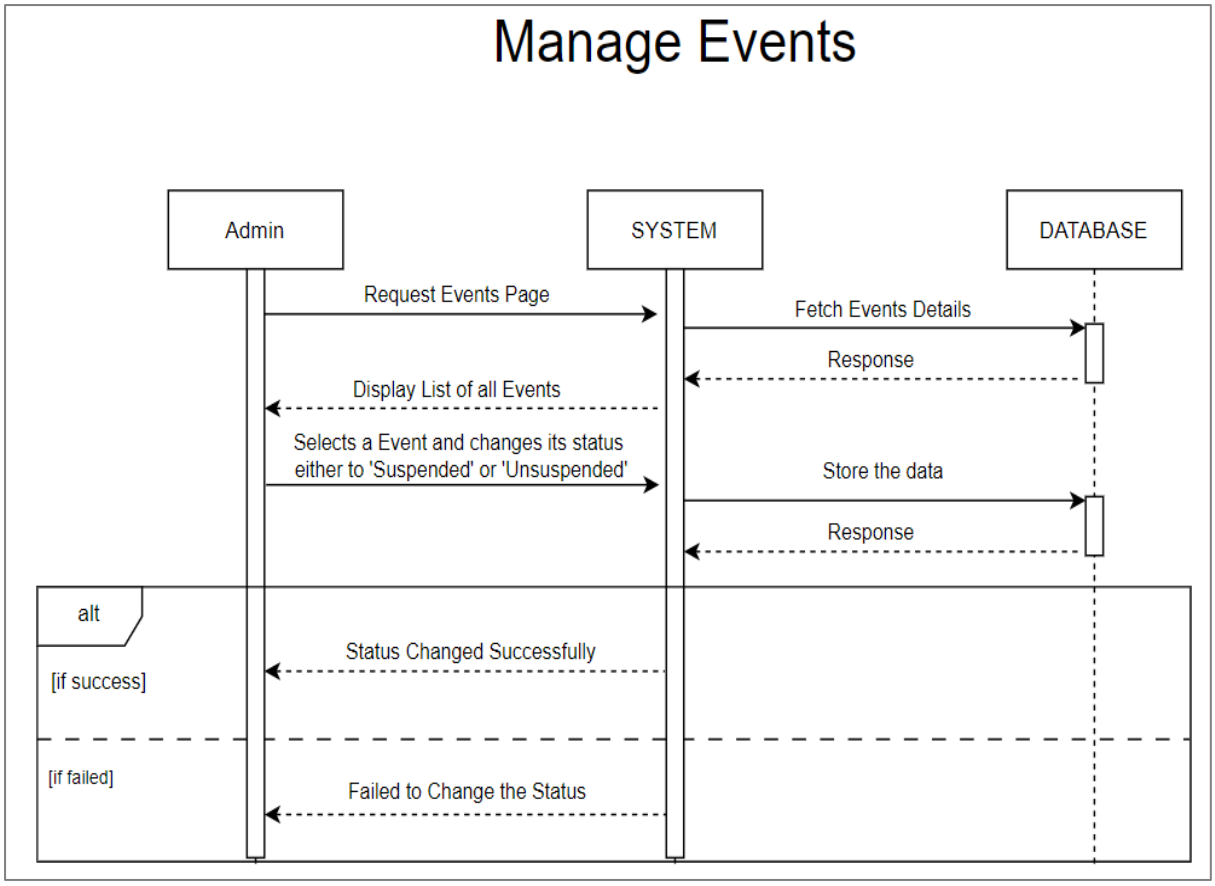
Figures 4.60 Sequence Diagram – Manage Application For Events



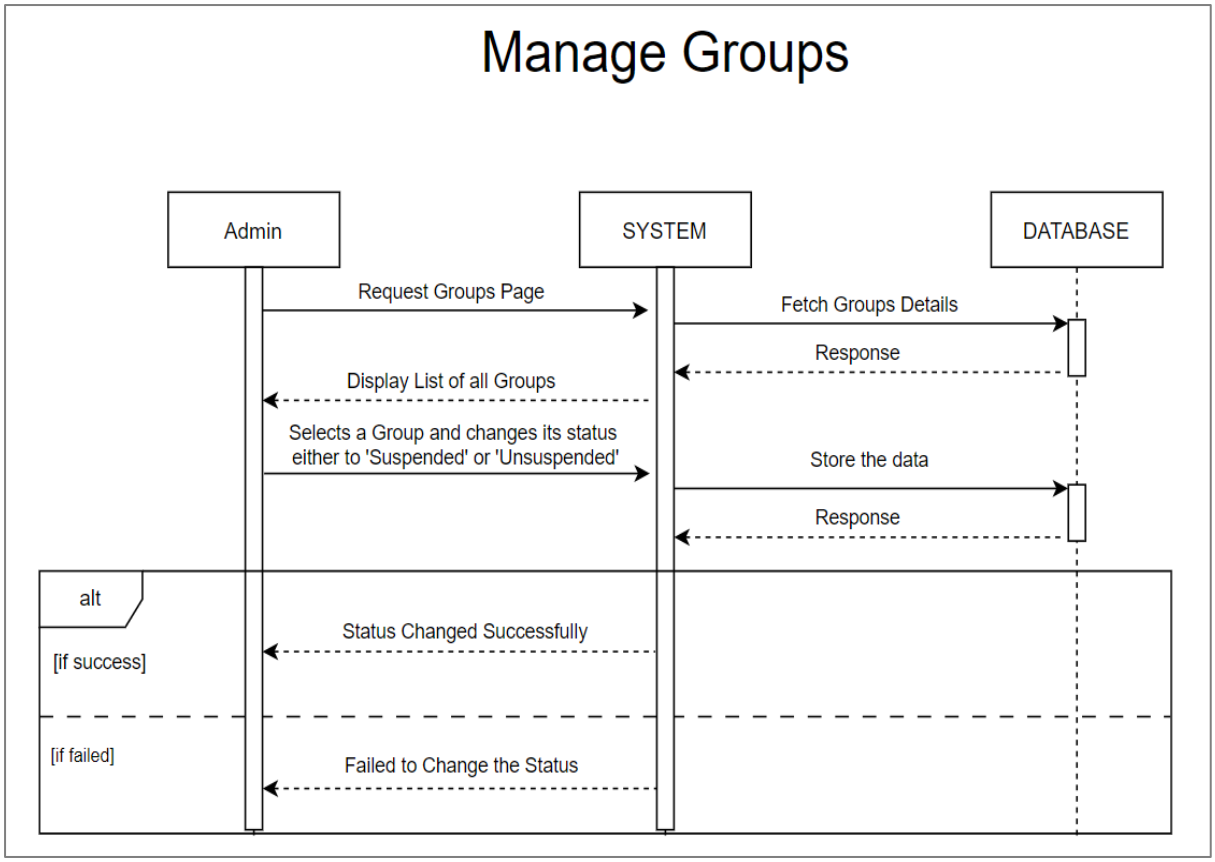
Figures 4.61 Sequence Diagram – Add Admin



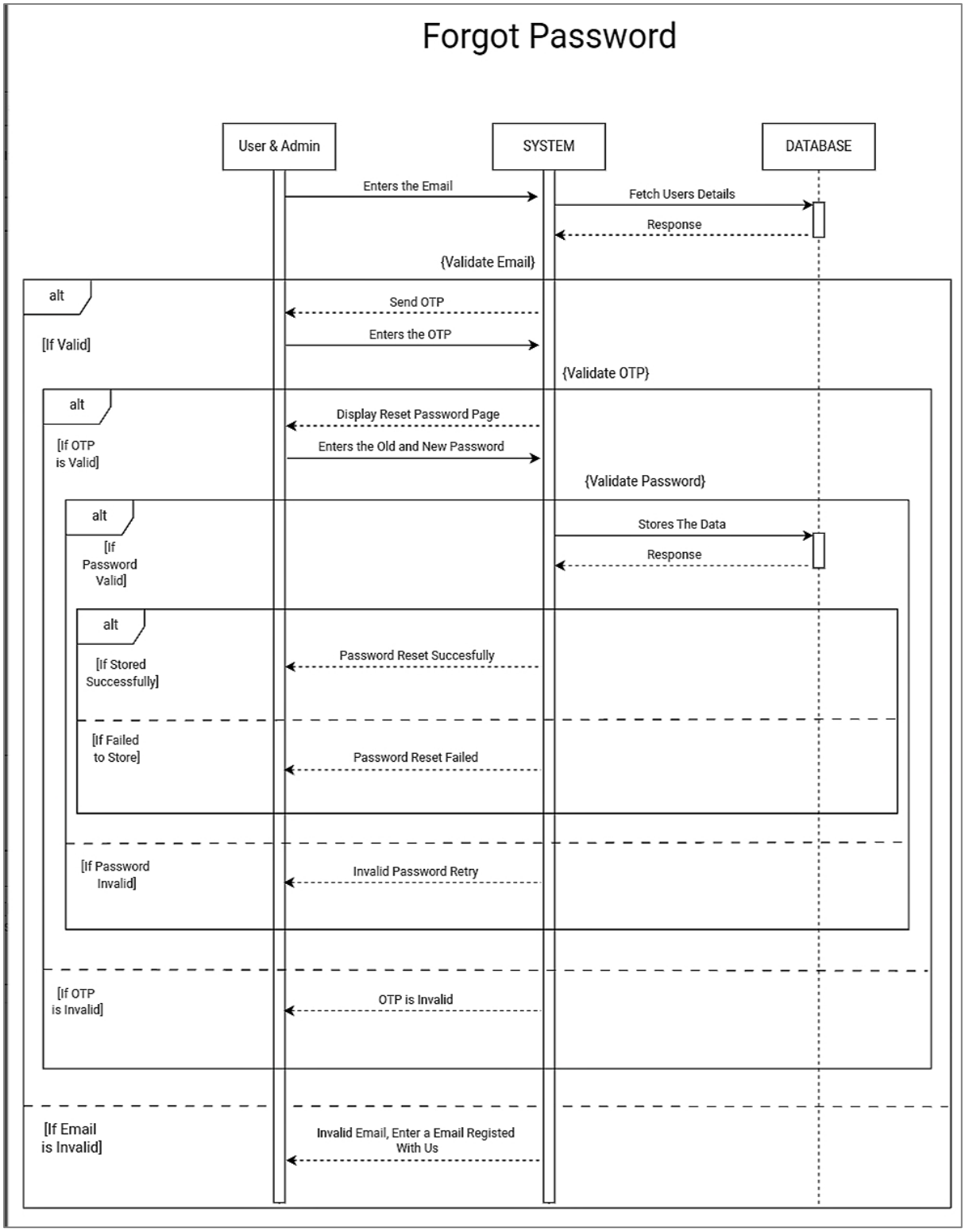
Figures 4.62 Sequence Diagram – Manage Users



Figures 4.63 Sequence Diagram – Manage Events



Figures 4.64 Sequence Diagram – Manage Groups



Figures 4.65 Sequence Diagram – Forgot Password

4.8 Activity Diagram:

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow

is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc

Diagram Notations:

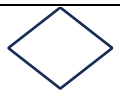
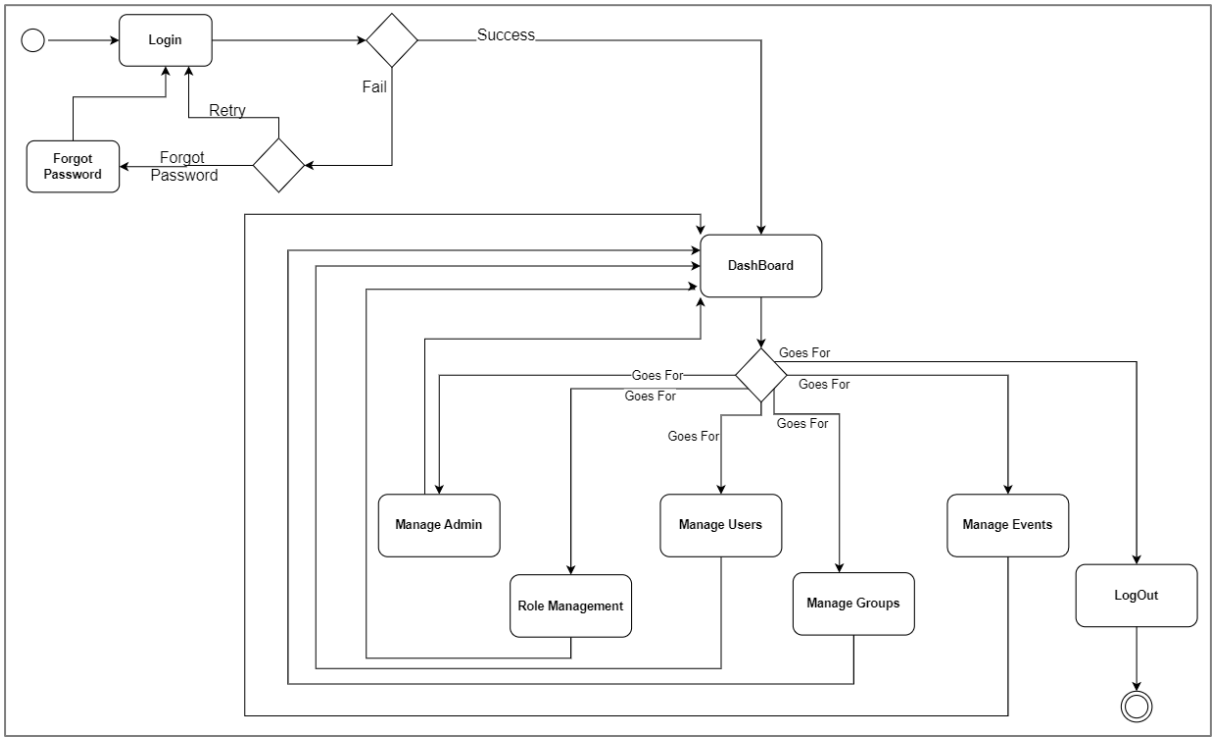
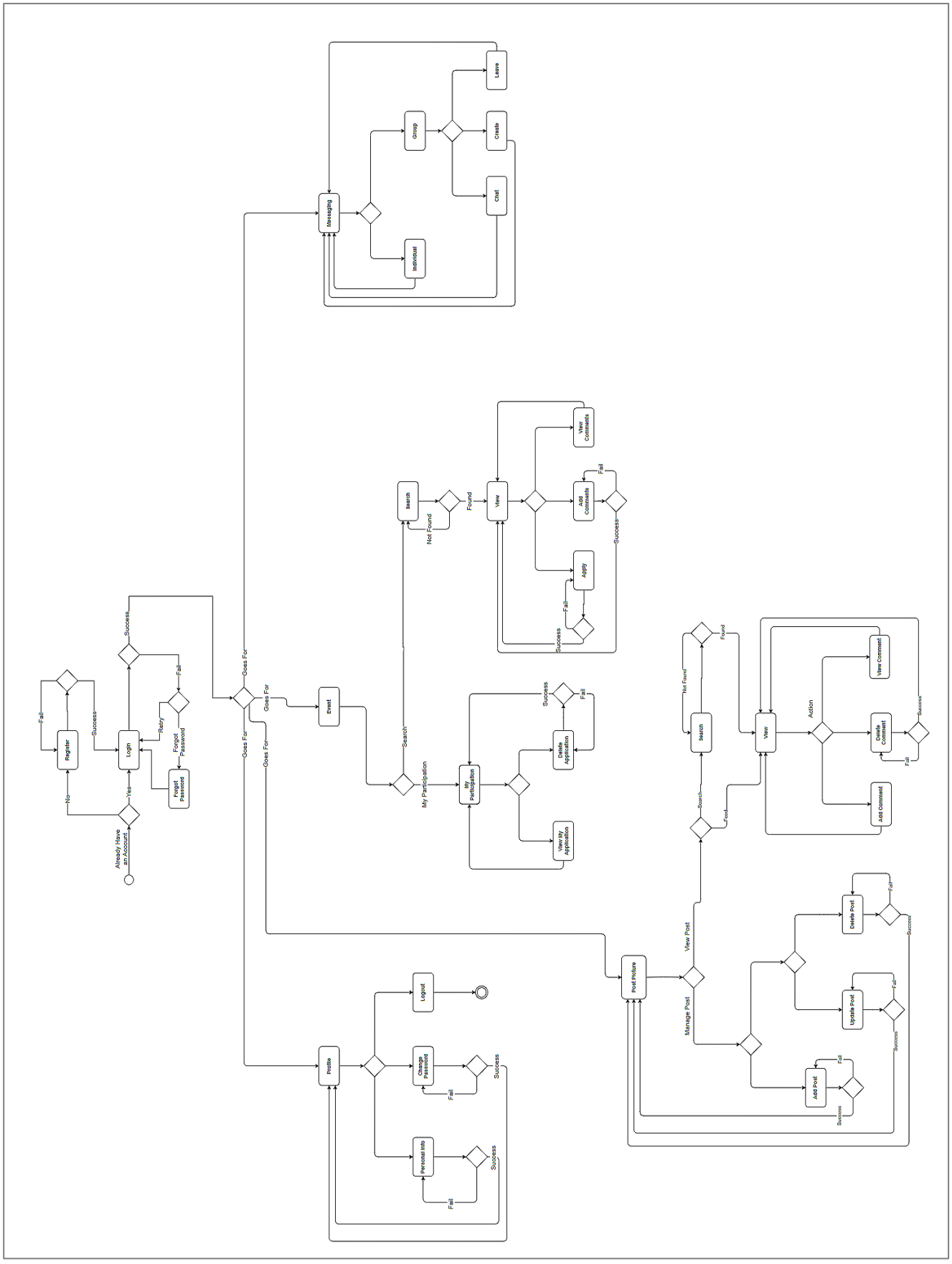
Name	Symbol	Description
Initial State		This shows the starting point or first activity of the flow.
Final State		The end of the Activity diagram, also called as a final activity.
Action		It represents the activity to be performed.
Decision		A logic where a decision is to be made is depicted by a diamond.
Transition		A transition link represents control flow between nodes.

Table 4.6 Activity Diagram notations

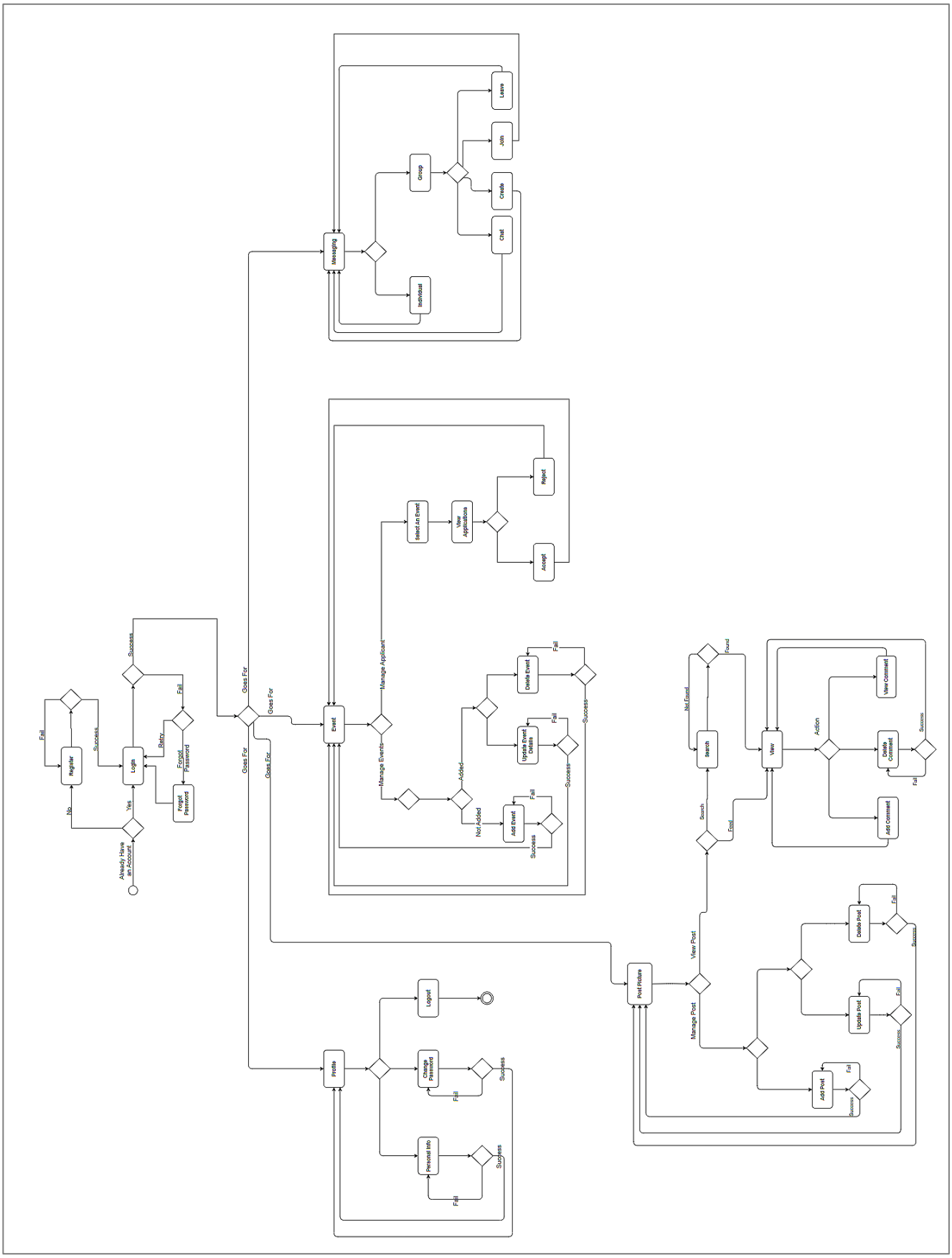
Diagram:



Figures 4.66 Activity Diagram of Admin



Figures 4.67 Activity Diagram of Volunteer



Figures 4.67 Activity Diagram of Organization

4.9 User Interface (UI) Design:

4.10 Test Case Design:

Test Case No	Test Case	Expected Output	Actual Output	Remark
1.				