Contents

[1. EXECUTIVE SUMMARY 3](#_Toc138791647)

[2. BACKGROUND 4](#_Toc138791649)

[I. Existing System 4](#_Toc138791650)

[II. Definition of problem 4](#_Toc138791651)

[III. Proposed System 4](#_Toc138791652)

[3. PROJECT OVERVIEW 5](#_Toc138791663)

[I. Objective of the Project 5](#_Toc138791664)

[II. Stakeholders 5](#_Toc138791665)

[III. Scope of the Project 6](#_Toc138791666)

[IV. Feasibility Analysis 6](#_Toc138791667)

[4. OVERALL PROJECT PLANNING 7](#_Toc138791668)

[I. Development Environment 7](#_Toc138791669)

[II. Constraints 8](#_Toc138791670)

[III. Deliverables 8](#_Toc138791671)

[IV. Assumptions and dependencies 8](#_Toc138791672)

[V. Risks 8](#_Toc138791673)

[VI. Process Model 9](#_Toc138791674)

[VII. Test Strategy 9](#_Toc138791675)

[VIII.Testing environment and tools 11](#_Toc138791676)

[5. ITERATION PLANNING 12](#_Toc138791677)

[I. Schedule (in weeks) 12](#_Toc138791678)

[II. Risks 12](#_Toc138791679)

[6. HIGH LEVEL SYSTEM ANALYSIS 12](#_Toc138791680)

[I. User Characteristics 12](#_Toc138791681)

[II. Summary of System Features/ Functional requirements 13](#_Toc138791682)

[III. Non Functional requirements/Supplementary Specifications 14](#_Toc138791683)

[IV. Glossary 14](#_Toc138791684)

[V. Business Rules 15](#_Toc138791685)

[VI. Use cases 15](#_Toc138791686)

[VII. Use case Diagram 17](#_Toc138791687)

[7. DOMAIN MODEL 19](#_Toc138791688)

[8. USE CASE MODEL 19](#_Toc138791690)

[I. Use case text 19](#_Toc138791691)

[II. System Sequence diagram 22](#_Toc138791692)

[III. Operation contracts 25](#_Toc138791693)

[IV. Reports 27](#_Toc138791704)

[V. Sequence diagrams 30](#_Toc138791705)

[VI. Class diagram 33](#_Toc138791712)

[VII. UI design 34](#_Toc138791714)

[VIII. Theoretical Background 44](#_Toc138791756)

[IX. Architecture 45](#_Toc138791757)

[X. Database design 47](#_Toc138791764)

[9. TESTING 50](#_Toc138791775)

[I. Test Cases 50](#_Toc138791776)

[II. Test Report 51](#_Toc138791798)

[III. Sample Code used for testing 53](#_Toc138791820)

[10. TRANSITION 54](#_Toc138791821)

[I. SYSTEM IMPLEMENTATION 54](#_Toc138791822)

[II. SYSTEM MAINTENANCE 55](#_Toc138791823)

[III. USER/OPERATIONAL MANUAL 56](#_Toc138791824)

[11. ANNEXURE 57](#_Toc138791841)

[I. Document Glossary 57](#_Toc138791842)

[II. References: 57](#_Toc138791851)

[III. User Interview Questionnaires: 58](#_Toc138791852)

[IV. Sample Project code: 59](#_Toc138791853)

# EXECUTIVE SUMMARY

The Volunteer Management system is a comprehensive platform designed to streamline the operations of non-profit organizations and community groups. It facilitates efficient management of volunteers and organizations, enabling them to collaborate effectively and create positive social impact.

### Users:

The system consists of three main user roles: Admin, Organizations, and Volunteers.

**Admin:**

• The Admin is responsible for overseeing the system and has access to various administrative functions.

• They can log in to the system securely.

• They review and approve or reject applications from organizations and volunteers.

• The Admin can view detailed information about organizations, volunteers, and events.

• The Admin can remove volunteers and organizations.

• The Admin can search for volunteers and organizations.

• They have the authority to accept or reject event requests from organizations.

• The Admin can generate reports on various aspects, such as the total number of volunteers, organizations, volunteer contributions, events conducted during specific time periods, total remuneration, volunteer intensive event, changes in volunteer compensation preference and donations received during specific time periods.

**Organizations:**

• Organizations can log in to their accounts and manage their information.

• They can view and edit their details, ensuring accurate representation.

• Organizations can add new events to the system, including relevant information and requirements.

• They have the ability to manage events efficiently, overseeing volunteer participation and coordinating logistics.

• Organizations can manage volunteering activities, ensuring smooth operations and effective communication with volunteers.

• They can compensate volunteers by providing payments for their services.

• Additionally, organizations can request new events, expanding their reach and impact.

• Organizations can generate reports on various aspects, such as the total number of volunteer applications, volunteer contribution, volunteer-intensive event, total amount of donation received during specific time periods, total remuneration and changes in volunteer compensation preference.

**Volunteers:**

• Volunteers can log in to their accounts to access personalized features.

• They can view and edit their personal details to keep their profiles up to date.

• Volunteers can specify their interests, allowing the system to recommend relevant events.

• The system presents volunteers with events based on their preferences and age.

• Volunteers can apply to volunteer for specific events, expressing their interest and availability.

• They can view the status of their volunteer activities to stay informed and organized.

• Volunteers have the option to donate to the organization, contributing to their cause.

• Organizations can provide payments to volunteers for their dedicated service.

• Volunteers can cancel their applications.

• Lastly, volunteers can download certificates of their volunteering activities, recognizing their efforts and achievements.

The Volunteer Management system empowers non-profits and community groups to efficiently manage volunteers, events, and collaborations. It enhances communication, simplifies administrative tasks, and promotes the overall effectiveness and impact of the organizations. By leveraging this platform, non-profits and community groups can foster meaningful relationships with volunteers, drive positive change, and achieve their missions more effectively.

# BACKGROUND

### Existing System

The existing system for volunteer management in non-profits and community groups is characterized by manual and disjointed processes. Organizations typically rely on spreadsheets, emails, and paper-based methods to manage volunteers and events. This fragmented approach leads to inefficiencies, communication gaps, and difficulty in tracking and coordinating volunteer activities. The lack of a centralized system hampers the organizations' ability to engage volunteers effectively, manage events efficiently, and generate comprehensive reports. As a result, there is a need for a more streamlined and integrated solution to overcome these challenges and optimize volunteer management processes.

### Definition of problem

The current volunteer management practices in non-profits and community groups present several challenges that hinder their effectiveness and impact. The absence of a centralized system leads to inefficiencies, miscommunication, and difficulty in tracking and coordinating volunteer activities. This manual and fragmented approach makes it challenging for organizations to engage volunteers effectively, manage events efficiently, and generate comprehensive reports. As a consequence, organizations struggle to maximize volunteer participation, streamline operations, and demonstrate the outcomes of their efforts. Therefore, there is a pressing need for a new Volunteer Management system to provide a cohesive and user-friendly platform that addresses these issues, enhances collaboration, and improves overall organizational performance.

### Proposed System

The proposed Volunteer Management system aims to address the limitations of the existing system by providing a centralized platform for non-profits and community groups. This system will streamline volunteer management processes, facilitate communication, and enhance overall operational efficiency. It will offer features such as user-friendly interfaces for admin, organizations, and volunteers, secure login functionality, application approval workflows, event management tools, volunteer activity tracking, donation management, and report generation capabilities. The proposed system will provide a cohesive and integrated solution to manage volunteers, events, and administrative tasks, resulting in improved collaboration, enhanced volunteer engagement, and better organizational outcomes.

Advantages of proposed system :

### Streamlined Operations

### Efficient Communication.

### Enhanced Volunteer Engagement

### Improved Event Management

### Simplified Administrative Tasks

### Better Reporting and Insights

### Secure and Accessible Platform

### Volunteer Recognition

### Financial Management

### Scalability and Growth

# PROJECT OVERVIEW

### Objective of the Project

The objective of the project is to develop a comprehensive Volunteer Management system that streamlines the operations of non-profits and community groups. The system aims to provide a centralized platform for effective management of volunteers, events, and administrative tasks. By offering user-friendly interfaces, seamless communication channels, and automated processes, the project aims to enhance volunteer engagement, improve event management, simplify administrative tasks, and generate valuable insights through comprehensive reporting. Ultimately, the objective is to empower organizations to drive positive social impact and achieve their missions more efficiently.

### Stakeholders

Admin:

The admin is responsible for managing the system, including approving/rejecting applications, viewing details of organizations/volunteers/events, removing organizations/volunteers, searching for organizations/volunteers, accepting/rejecting event requests, and generating reports.

Organizations:

Organizations can log in, view/edit their details, add/manage events, manage volunteering activities, pay volunteers, request new events, and communicate effectively with volunteers.

Volunteers:

Volunteers can log in, view/edit their personal details and interests, browse and apply for events, track their activity status, donate to organizations, receive payment for volunteering, and download certificates of their volunteering activities.

Top of Form

### Scope of the Project

The scope of the Volunteer Management system includes comprehensive features for managing volunteers, organizations, and events within non-profits and community groups. It encompasses user authentication, application processing, profile management, event creation and coordination, volunteer assignment tracking, communication tools, donation management, payment processing, and report generation. The system aims to streamline operations, optimize volunteer engagement, simplify administrative tasks, and provide valuable insights through reporting functionalities. It is designed to be scalable and adaptable to meet the evolving needs of organizations and support their mission-driven initiatives effectively.

### Feasibility Analysis

* 1. Technical Feasibility:

The Volunteer Management system developed in PHP using XAMPP and hosted on localhost is technically feasible. The chosen technology stack provides the necessary tools and infrastructure to implement the system's functionalities effectively. PHP is a widely used programming language for web development, while XAMPP provides a comprehensive local server environment. The system can utilize databases for data storage, handle user authentication, implement event management features, and facilitate secure communication between users.

* 1. Operational Feasibility:

The operational feasibility of the Volunteer Management system is ensured by its ability to address the operational needs and requirements of non-profits and community groups. The system streamlines volunteer management processes, enhances communication, and simplifies administrative tasks. By offering user-friendly interfaces and automation capabilities, it increases operational efficiency, improves volunteer engagement, and enables effective event coordination. The system is designed to meet the practical operational needs of organizations, making it operationally feasible.

* 1. Schedule Feasibility:

The development timeline of three months for the Volunteer Management system using PHP and XAMPP is considered feasible. The chosen technology stack allows for efficient web development, and the project can be broken down into smaller tasks and milestones to ensure progress within the given timeframe. Adequate planning, resource allocation, and efficient task management can help meet the development objectives and deliver the system within the set schedule.

The current schedule is depicted below which shows the time taken to complete each stages: (in weeks)

|  |  |  |
| --- | --- | --- |
| Problem Identification | – | 1 |
| Requirement Analysis | – | 2 |
| Overall Design | – | 3 |
| Construction | – | 4 |
| Testing | – | 2 |

* 1. Economic Feasibility – Cost – Benefit Analysis:

The development and implementation of the Volunteer Management system using PHP and XAMPP on a localhost are economically feasible. The open-source nature of PHP and XAMPP eliminates licensing costs, reducing the overall project expenses. Hosting the system on a localhost also eliminates the need for external hosting services, further reducing operational costs. Additionally, the system's potential benefits, such as improved volunteer management, streamlined operations, and enhanced impact, justify the investment and make it economically viable for non-profits and community groups.

# OVERALL PROJECT PLANNING

### Development Environment

#### Hardware Specifications

Processor : Pentium 4 or above

Memory : 2GB, 4 GB recommended

Display : Colour Monitor

Keyboard : Windows compatible

Mouse : Windows compatible

#### Software Specifications

* 1. Server Side:

Front end : HTML, CSS

Back end : MySQL, PHP Operating System : Windows

* 1. Client side:

Operating system : Windows

Software packages : Web browser (Ex: Chrome, Safari etc...)

### Constraints

* The user interface is available only in English, which may limit accessibility for non-English-speaking users.
* Limited to HTTP/HTTPS
* Additional security measures are necessary to protect user data and prevent unauthorized access.
* The system may need to integrate with other existing systems or platforms, requiring compatibility and data synchronization considerations.
* The system should be designed to handle increased user, organization, and event volumes as it grows over time.
* Regular maintenance and updates are required to address bugs and ensure optimal system performance.

### Deliverables

A User manual will be provided as a deliverable. It’s contents include,

* Feature Document
* Source code
* Database tables and structure
* Database Backup
* System Maintenance Document

### Assumptions and dependencies

* End user should have basic knowledge of computer
* All necessary hardware and software are available for implementing and use of the tool
* The code should be free of compilation errors/syntax errors.
* The product must have an interface which is simple enough to understand.

### Risks

Risk is any unexpected event that can affect your project — for better or for worse. Risk can affect anything: people, processes, technology, and resources. Some of them are:

* Backend files getting corrupted
* Database crashes
* External attacks from hackers

### Process Model

Agile methodology was used throughout the development of this system as it is much more flexible than other popular models. The phases include:

Phase 1: Requirements.

Phase 2: Design.

Phase 3: Development and Coding.

Phase 4: Integration and Testing.

Phase 5: Implementation and Deployment.

Phase 6: Review.

The overall goal of each Agile method is to adapt to change and deliver working software as quickly as possible. Most of the phases in the system were developed simultaneously, under the convenience of the developers. Many previous phases could only be developed after the later ones, which is extremely handy.

### Test Strategy

**Unit Testing:**

Unit testing is an essential component of the software development process. It involves testing individual units or components of the system to ensure they function correctly in isolation. In our Volunteer Management system, we conducted unit testing on various modules and functions within the codebase. Each function was tested with different input values to verify its expected behavior and output. By isolating and testing individual units, we were able to identify and fix any issues at an early stage, ensuring the overall stability and reliability of the system.

**Module Testing:**

Module testing focuses on testing groups of related units or modules that work together to achieve specific functionality. In our Volunteer Management system, we performed module testing to verify the interactions and compatibility between different components. We tested the integration of modules such as user authentication, event management, and volunteer tracking. This allowed us to detect any integration issues or dependencies that may have affected the system's overall performance. By thoroughly testing the modules, we aimed to ensure the seamless integration of different parts of the system.

**Integration Testing:**

Integration testing is conducted to validate the proper functioning and communication between different modules or subsystems of the system. In our Volunteer Management system, we performed integration testing to verify that the various components, such as the front-end interface and back-end functionalities, were working together correctly. We tested the flow of data and information between different user roles, such as admins, organizations, and volunteers. By identifying and resolving any integration issues, we aimed to ensure the smooth operation of the entire system.

**Validation Testing:**

Validation testing focuses on verifying that the system meets the specified requirements and satisfies the needs of the stakeholders. In our Volunteer Management system, we conducted validation testing to ensure that the implemented features and functionalities aligned with the desired objectives. We worked closely with the stakeholders, including non-profit organizations and community groups, to gather feedback and validate that the system met their expectations. Through user acceptance testing and continuous feedback loops, we were able to validate the system's effectiveness and make necessary adjustments.

**Acceptance Testing:**

Acceptance testing is performed to determine whether the system meets the acceptance criteria defined by the stakeholders. In our Volunteer Management system, we collaborated with the stakeholders, including admins, organizations, and volunteers, to conduct acceptance testing. We created test scenarios and performed end-to-end testing to ensure that the system fulfilled the requirements and provided a satisfactory user experience. By involving the stakeholders in the testing process, we aimed to obtain their final approval and ensure that the system was ready for deployment.

**System Testing:**

System testing involves testing the entire system as a whole, including all its integrated components and functionalities. In our Volunteer Management system, we performed system testing to validate the overall behavior and performance of the application. We tested various user interactions, system responses, and error handling scenarios to ensure that the system functioned as expected in different usage scenarios. By executing comprehensive system tests, we aimed to identify any issues that may arise from the interaction between different components or modules.

**Black Box Testing:**

Black box testing involves testing the system's external behavior without any knowledge of its internal structure or implementation. In our Volunteer Management system, we conducted black box testing to evaluate the system's functionality from a user's perspective. We created test cases based on the system's requirements and tested the inputs and outputs to ensure proper system behavior.

**White Box Testing:**

White box testing, also known as structural or code-based testing, involves examining the system's internal structure and implementation. In our Volunteer Management system, we performed white box testing to ensure code quality, assess code coverage, and identify any potential defects or vulnerabilities. By reviewing the code and executing targeted tests, we aimed to validate the reliability and robustness of the system's implementation.

Overall, through rigorous testing practices including unit testing, module testing, integration testing, validation testing, acceptance testing, system testing, and both black and white box testing, we ensured the quality and reliability of our Volunteer Management system. By conducting thorough tests at each level, we were able to identify and address any issues or defects early in the development cycle, minimizing risks and improving the overall performance of the system. These testing processes allowed us to verify the functionality of individual components, test their integration, validate the system against requirements, gain user acceptance, and ensure the system's robustness and security. The comprehensive testing approach played a crucial role in delivering a high-quality, stable, and user-friendly Volunteer Management solution for non-profit organizations and community groups.

### Testing environment and tools

Hardware Specifications

Operating System – Windows 10 Memory – 16GB

Hard Disk – 1TB

CPU – Intel I5 9th Gen

Software Specifications

Tools – VS Code

Front end – HTML, CSS, PHP

Back end – MySQL, PHPMYADMIN

Internet Standard – HTTP, HTTPS Operating System – Windows 10

# ITERATION PLANNING

### Schedule (in weeks)

|  |  |
| --- | --- |
| Problem Identification | – 1 |
| Requirement Analysis | – 2 |
| Overall Design | – 3 |
| Construction | – 4 |
| Testing | – 2 |
| Total | – 12 weeks = 84 days |

### Risks

* + The system relies on stable internet connectivity for users to access and utilize its features effectively.
  + There is a risk of unauthorized access, data breaches, or misuse of personal information stored within the system.
  + Delays in the admin's approval of organization and volunteer applications can result in user frustration and disengagement.
  + There is a risk of low user adoption and engagement if the system is not user-friendly, lacks adequate features, or fails to meet the needs of the organizations and volunteers.

# HIGH LEVEL SYSTEM ANALYSIS

### User Characteristics

Users are expected to have basic knowledge of using a computer and basics of English as it’s the only language available.

Users of the system:

* + Admin
  + Organizations
  + Volunteers

### Summary of System Features/ Functional requirements

Main modules include,

**User Management Module:**

* Registration and login functionality for volunteers and social organizations.
* User profile management, including personal details and preferences.
* Verification for user registration by the admin.

**Event Management Module:**

* Social organizations can create and post events on the platform.
* Events can include details such as event type, date, time, location, and required number of volunteers.
* Volunteers can view events based on their preferences and criteria set by organizers (e.g., age restrictions).
* Volunteer applications are accepted until the required number of volunteers is reached.

**Payment Module:**

* Social organizations can set a nominal payment for volunteering activities, which is displayed in the event description.
* Volunteers can choose to receive the payment or donate it to the organization.
* The module handles the payment process securely and tracks the transaction history.
* Donation funds are added to the organization's balance.

**Reporting Module:**

* Generates a report on the total number of volunteer activities.
* Provides a report on the number of volunteers willing to donate.
* Generates a report on the amount of money received by each social organization through volunteering activities.
* Generates a report on total number of events and volunteer applications for each organization.
* Generates a report on volunteer-intensive event
* Provides a report on changes in volunteer compensation preference
* Helps in evaluating the impact of volunteering activities through generated reports.
* Assists in assessing the platform's effectiveness by providing reports.

**Request Module:**

* Allows social organizations to request the admin to add new events to the platform.
* The admin can review and approve or reject event requests based on the platform's requirements and quality standards.

**Certificate Module:**

* Generates personalized certificates for volunteers, highlighting their contribution to a specific event
* Certificates serve as recognition of the volunteer's efforts and can be used for professional or educational purposes.

### Non Functional requirements/Supplementary Specifications

Scalability:

Scalability is a crucial non-functional requirement that focuses on the system's ability to handle increasing workloads and accommodate a growing number of users, organizations, and volunteers. The system should effectively scale up its resources to ensure optimal performance without significant degradation as the usage grows.

Reliability:

Reliability emphasizes the system's consistency in performing its specified functions without failure. It measures the system's uptime and the mean time between failures (MTBF). The Volunteer Management system should exhibit high reliability, minimizing the occurrence of errors or failures and ensuring uninterrupted operation for users.

Maintainability:

Maintainability refers to the system's ease of support, changes, enhancements, and restructuring over time. It focuses on making the system easily manageable for future updates and modifications. The Volunteer Management system should have well-structured, modular code that is properly documented, enabling efficient maintenance and reducing downtime during updates or improvements.

Usability:

Usability addresses the system's ease of use for volunteers and social organizations. It aims to provide an intuitive and user-friendly interface, allowing users to navigate and interact with the system effortlessly. Clear and intuitive forms, along with appropriate input validation, should be implemented to ensure a positive user experience and efficient utilization of system features.

Interoperability:

Interoperability refers to the system's capability to share information and exchange data with other systems and external hardware seamlessly. The Volunteer Management system should support interoperability by utilizing standardized data formats, transport protocols, and interfaces. This enables easy information exchange and integration with other systems, enhancing collaboration and data sharing between different entities.

### Glossary

**Volunteer Management System**: Platform for managing volunteers and organizations in non-profit and community groups.

**Admin**: System manager responsible for approval, details, removal, search, event handling, and reporting.

**Organization**: Non-profit or community group managing volunteer activities.

**Volunteer**: Individual providing unpaid work for organizations.

**Event**: Organized activity involving volunteers.

**Application Approval**: Process of reviewing and accepting/rejecting applications.

**Activity Status**: Volunteer's progress in an event (pending, approved, completed).

**Remuneration**: Payment provided to volunteers for their duties.

**Donation**: Voluntary contribution supporting organization's activities.

**Certificate**: Document recognizing volunteer's participation in an event.

### Business Rules

The information should be correct and valid.

### Use cases

**User Management:**

* Admin can register a new user (volunteer or organization).
* Admin can approve or reject user applications.
* Admin can view user details.
* Users can log in to their accounts.

**Event Management:**

* Organizations can create new events.
* Organizations can edit event details.
* Organizations can delete events.
* Organizations can manage volunteer applications for events.
* Volunteers can view events based on their preferences and age .
* Volunteers can apply to volunteer for specific events.

**Payment Management:**

* Organizations can set a payment amount for volunteering activities.
* Volunteers can choose to receive payment or donate it to the organization.
* Payments are processed securely.

**Reporting:**

* Admin can generate reports on the total number of volunteers, organizations, events and volunteer contributions
* Admin can generate report on the total amount received through donation
* Admin can generate reports on events and donations during a specific time period.
* Admin can generate report on total remuneration
* Admin can generate report on volunteer intensive event
* Admin can generate report on changes in volunteer compensation preference
* Organization can generate reports on the total number of events conducted and volunteers applied.
* Organization can generate report on the total amount received through donation
* Organization can generate report on donations during a specific time period.
* Organization can generate report on total remuneration
* Organization can generate report on volunteer intensive event
* Organization can generate report on changes in volunteer compensation preference

**Event Request:**

* Organizations can request the admin to add new events.
* Admin can review and approve or reject event requests.

**Certificate Generation:**

* Admin can generate personalized certificates for volunteers, highlighting their participation in specific events.

**Volunteer Profile Management:**

* Volunteers can view and edit their personal details.
* Volunteers can view and edit their interests.

**Volunteer Activity Status:**

* Volunteers can view the status of their volunteer activities.
* Volunteers can track their participation and progress.

**Organization Profile Management:**

* Organizations can view and edit their details.

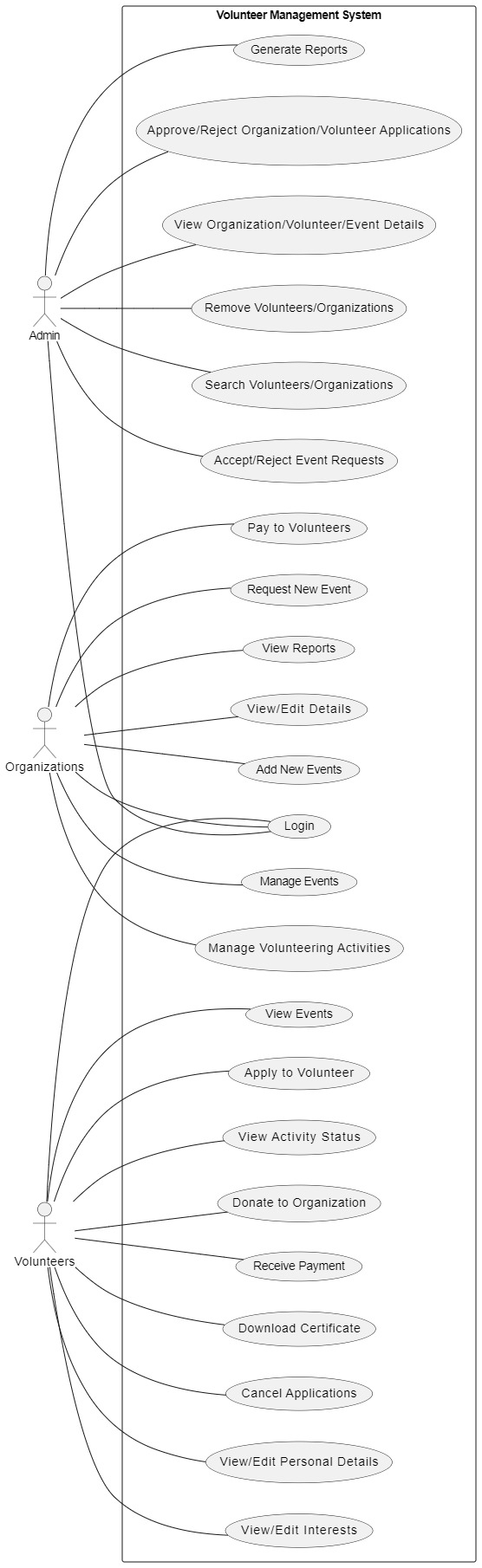
**Volunteer-Organization Interaction:**

* Volunteers can donate to organizations.
* Organizations can pay volunteers for their volunteering duties.

**Certificate Download:**

* Volunteers can download certificates of their volunteering activities.

### Use case Diagram

****

# DOMAIN MODEL

# 

# USE CASE MODEL

### Use case text

**Scope:** The use case involves the management of volunteers, organizations, and events for non-profit and community groups.

**Primary Actors:**

Admin:

Responsible for overall system administration and management.

Organizations:

Non-profit organizations and community groups utilizing the system.

Volunteers:

Individuals interested in volunteering for various events and activities.

**Stakeholder and interests:**

Admin:

Interested in efficient management of user registrations, event approvals, and generating reports.

Organizations:

Interested in managing their profile, creating and managing events, and tracking volunteer activities.

Volunteers:

Interested in viewing and applying for volunteer opportunities, managing personal details, and tracking their activity status.

##### Special Requirements:

There are no special requirements for this use case.

##### Preconditions:

Admin, organizations, and volunteers must have valid user credentials to access the system.

##### Success Guarantee (Post conditions):

The system provides a user-friendly and efficient platform for managing volunteers, organizations, and events.

##### Main Success Scenario:

Admin:

* Logs in to the system using valid credentials.
* Approves or rejects organization and volunteer applications.
* Views details of organizations, volunteers, and events.
* Accepts or rejects event requests from organizations.
* Generates reports on volunteer, organization, event, donation, remuneration and compensation preference statistics.

Organizations:

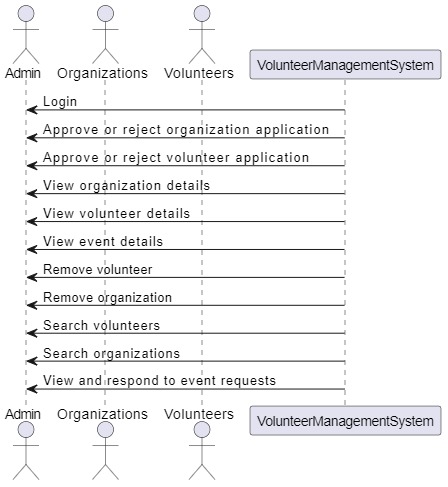
* Logs in to the system using valid credentials.
* Views and edits their profile details.
* Adds new events to the system.
* Manages events by editing or deleting them.
* Manages volunteering activities by reviewing volunteer applications.
* Pays volunteers for their services.
* Requests the addition of new events through the admin.
* Generates reports on events, volunteers, donation, remuneration and compensation preference statistics.

Volunteers:

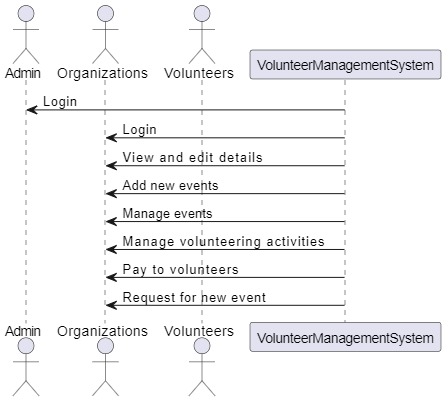
* Logs in to the system using valid credentials.
* Views and edits their personal details.
* Views their interests and can make changes.
* Views events based on their preferences and age criteria.
* Applies to volunteer for specific events.
* Tracks the status of their volunteer activities.
* Donates to organizations.
* Receives payment from organizations for their volunteering duties.
* Downloads certificates of their volunteering activities.

### System Sequence diagram

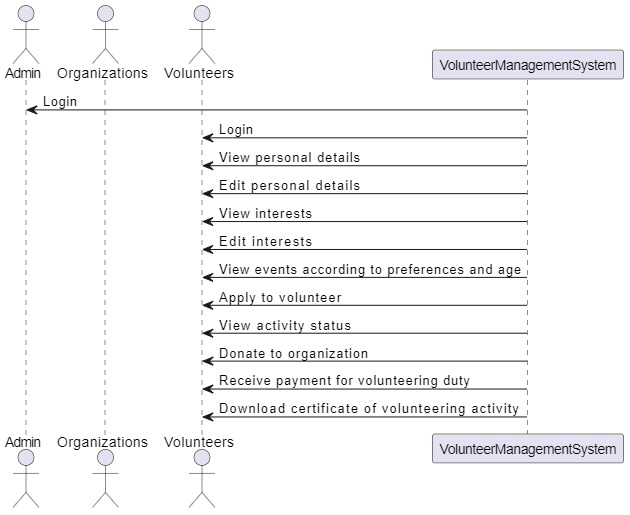
Admin



Social Organizations



Volunteers



### Operation contracts

**Admin**

Operation: Login

Cross References: Admin

Preconditions:

The Volunteer Management System is running and accessible.

The admin user has a valid account in the system.

Postconditions:

The admin user is successfully logged in to the system.

The admin user can access the admin functionalities.

Flow of Events:

The admin user launches the Volunteer Management System.

The system presents the login screen.

The admin user enters their credentials (username and password) and submits the login form.

The system verifies the admin user's credentials.

If the credentials are valid:

The system authenticates the admin user and grants access to admin functionalities.

If the credentials are invalid, an error message is displayed, and the admin user is prompted to re-enter the correct credentials.

Exceptions:

If there are technical issues or the server is unreachable, an error message is displayed, and the admin user is prompted to try again later.

**Social Organizations**

Operation: Login

Cross References: Organizations

Preconditions:

The Volunteer Management System is running and accessible.

The organization user has a valid account in the system.

Postconditions:

The organization user is successfully logged in to the system.

The organization user can access the functionalities specific to their role.

### Flow of Events:

The organization user launches the Volunteer Management System.

The system presents the login screen.

The organization user enters their credentials (username and password) and submits the login form.

The system verifies the organization user's credentials.

If the credentials are valid:

The system authenticates the organization user and grants access to organization-specific functionalities.

If the credentials are invalid, an error message is displayed, and the organization user is prompted to re-enter the correct credentials.

### Exceptions:

If there are technical issues or the server is unreachable, an error message is displayed, and the organization user is prompted to try again later.

This operation contract outlines the behavior of the login functionality in the Volunteer Management System for the Organizations.

**Volunteers**

### Operation: Login

### Cross References: Volunteers

### Preconditions:

The Volunteer Management System is running and accessible.

The volunteer user has a valid account in the system.

### Postconditions:

The volunteer user is successfully logged in to the system.

The volunteer user can access the functionalities specific to their role.

### Flow of Events:

The volunteer user launches the Volunteer Management System.

The system presents the login screen.

The volunteer user enters their credentials (username and password) and submits the login form.

The system verifies the volunteer user's credentials.

If the credentials are valid:

The system authenticates the volunteer user and grants access to volunteer-specific functionalities.

If the credentials are invalid, an error message is displayed, and the volunteer user is prompted to re-enter the correct credentials.

### Exceptions:

### If there are technical issues or the server is unreachable, an error message is displayed, and the volunteer user is prompted to try again later.

### This operation contract outlines the behavior of the login functionality in the Volunteer Management System for the Volunteers.

### Reports

The reports are generated according to specific category and input.

**Design Model**

#### UI Model

#### Admin login:

* + Login: The admin can login to access the features using this section. It is located on the homepage. The admin can enter his/her details which the system uses to authenticate admin. This is later on used in the login section
  + Dashboard: This is the default page the admin will land on after entering the system. Admin would have an interface to manage all the modules of the system. Admin can view reports
  + Organizations: This section provides details of registered organizations
  + Volunteers: This section provides details and interests of registered volunteers
  + Requests: This section provides details of event requests from organizations.
  + Events: This section generates report on events during specific time periods.
  + Donations: This section generates report on donations during specific time periods.
  + Preferences: This section generates report on volunteer compensation preference during specific time periods.
  + Organization applications: This section provides application details from various organizations.
  + Volunteer applications: This section provides application details from various volunteers.

#### Organization login:

* + Login: The organization can login to access the features using this section. It is located on the homepage. The admin can enter his/her details which the system uses to authenticate the organization. This is later on used in the login section
  + Home page: This is the default page the organization will land on after entering the system. Organization can view their details and would have an interface to interact with the system features.
  + Edit details: Organization can edit their details
  + Events: Organization can manage their events
  + Edit events: Organization can edit event details.
  + Volunteer applications: Organization can view applications from different volunteers.
  + Activities: Organization can manage volunteering activities.
  + Payment: Organization can pay for volunteering services.
  + New event: Organization can add new event.
  + Request: Organization can request for a new event.
  + Reports: Organizations can view reports.
  + Donations: This section generates report on donations during specific time periods.
  + Preferences: This section generates report on volunteer compensation preference during specific time periods.

#### Volunteer login:

* + Login: The volunteer can login to access the features using this section. It is located on the homepage. The volunteer can enter his/her details which the system uses to authenticate the volunteer. This is later on used in the login section
  + Home page: This is the default page the volunteer will land on after entering the system. Volunteers can view their personal details and would have an interface to interact with the system features.
  + Edit details: Volunteers can edit their personal details
  + Interests: Volunteers can view their interests
  + Edit interests: Volunteers can edit their interests.
  + Events: Volunteers can view, cancel and apply for events based on their preferences and age criteria.
  + Activities: Volunteers can view their activity status and download certificates.

### Sequence diagrams

### Admin

### 

### Social Organizations

### 

### Volunteers

### 

### Class diagram

### z

### UI design

### For Admin:

### Login screen for the admin

### 

### Dashboard or home screen for the admin

### 

### Applications management screen (to approve or reject organization/volunteer applications)

### 

### 

### Organizations details screen

### 

### Volunteers details screen

### 

### Events details screen

### 

### Reports screen (to generate reports on various metrics)

### 

### 

### For Organizations:

### Login screen for organizations

### 

### Home screen for organizations

### 

### Organization details screen (to view and edit organization details)

### 

### Add new event screen (to create and add new events)

### 

### Manage events screen (to view and manage existing events)

### 

### Reports screen (to generate reports on various metrics)

### 

### For Volunteers:

### Login screen for volunteers

### 

### Events screen (to view events based on preferences and age)

### 

### Personal details screen (to view and edit personal details)

### 

### 

### Apply to volunteer screen (to apply for volunteering)

### 

### Activity status screen (to view the status of volunteer activities)

### 

### Theoretical Background

The Volunteer Management system for Non-Profits and Community Groups is a web-based application that utilizes a combination of technologies to facilitate efficient volunteer and organization management. The system is built using a front-end framework consisting of HTML and CSS, providing a user-friendly interface for seamless interaction. The back-end functionality is implemented using PHP, which handles the server-side processing and data management. The system is deployed locally on the XAMPP server, ensuring a secure and reliable environment.

The theoretical background of this project lies in the concept of volunteer management and its importance in the non-profit sector. Volunteer management encompasses the processes and strategies employed by organizations to attract, recruit, train, and retain volunteers. It plays a vital role in optimizing the utilization of volunteer resources and achieving organizational objectives.

By developing a comprehensive Volunteer Management system, this project aims to address the challenges faced by non-profit organizations and community groups in effectively managing volunteers and organizations. The system streamlines administrative tasks and provides a centralized platform for efficient collaboration and communication among stakeholders.

The choice of HTML, CSS, and PHP technologies enables the development of a user-friendly and dynamic interface, ensuring a smooth user experience. The use of XAMPP as the server platform ensures a secure and reliable hosting environment for the system.

Overall, this project combines the theoretical principles of volunteer management with the practical application of web technologies to create an innovative solution that empowers non-profit organizations and community groups to optimize their volunteer management processes and create a positive social impact.

### Architecture

### Package diagram

### 

### Component diagram

### 

### Deployment diagram

### 

### Database design

### tbl\_admin

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.NO** | **Field** | **Type** | **Key** | **Description** |
| 1 | aid | int(10) | PRI | User Id of admin |
| 2 | aname | varchar(50) |  | Full name |
| 3 | aemail | varchar(100) |  | Email address |
| 4 | apswd | varchar(50) |  | Admin password |

### tbl\_organization

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.NO** | **Field** | **Type** | **Key** | **Description** |
| 1 | oidPrimary | int(10) | PRI | Id of organization |
| 2 | oname | varchar(100) |  | Name of the organization |
| 3 | ocontactname | varchar(50) |  | Contact person |
| 4 | ophone | varchar(20) |  | Contact number |
| 5 | oemail | varchar(100) |  | Email address |
| 6 | opswd | varchar(50) |  | Organization password |
| 7 | oaddr | varchar(100) |  | Address of the organization |
| 8 | odescription | varchar(200) |  | Description about the organization |
| 9 | obalance | int(10) |  | Account balance |
| 10 | ostatus | varchar(50) |  | Status of organization:  pending/approved/rejected |

### tbl\_volunteer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.NO** | **Field** | **Type** | **Key** | **Description** |
| 1 | vid | int(10) | PRI | User Id of volunteer |
| 2 | vname | varchar(25) |  | Name of the volunteer |
| 3 | vphone | varchar(20) |  | Contact number |
| 4 | vemail | varchar(100) |  | Email address |
| 5 | vpswd | varchar(50) |  | Volunteer password |
| 6 | vaddr | varchar(150) |  | Address of the volunteer |
| 7 | vage | int(10) |  | Age |
| 8 | vgender | varchar(10) |  | Gender |
| 9 | vstatus | varchar(50) |  | Status of volunteer:  pending/approved/rejected |
| 10 | vcard | blob |  | Stores the id proof of volunteers |

### 

### tbl\_eventtype

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.NO** | **Field** | **Type** | **Key** | **Description** |
| 1 | etypeid | int(10) | PRI | Id of event type |
| 2 | etypename | varchar(100) |  | Name of event type |

### tbl\_eventlist

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.NO** | **Field** | **Type** | **Key** | **Description** |
| 1 | elid | int(10) | PRI | Id of event name |
| 2 | etypeid | int(10) | FK | Event type |
| 3 | elname | varchar(100) |  | Event name |

### tbl\_event

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.NO** | **Field** | **Type** | **Key** | **Description** |
| 1 | eid | int(10) | PRI | Id of event |
| 2 | elid | int(10) | FK | Id of event name |
| 3 | oid | int(10) | FK | Id of organization |
| 4 | edescription | varchar(200) |  | Event description |
| 5 | estart | date |  | Start date |
| 6 | eend | date |  | End date |
| 7 | etime | varchar(50) |  | Time |
| 8 | elocation | varchar(100) |  | Event location |
| 9 | eagell | int(10) |  | Lower limit of age |
| 10 | eageul | int(10) |  | Upper limit of age |
| 11 | ereqno | int(10) |  | Number of volunteers required |
| 12 | eappliedno | int(10) |  | Number of volunteers applied |
| 13 | ewage | int(10) |  | Wage for each volunteer |
| 14 | estatus | varchar(10) |  | Status of the event:(complete/incomplete) |

### tbl\_request

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.NO** | **Field** | **Type** | **Key** | **Description** |
| 1 | rid | int(10) | PRI | Id of request |
| 2 | oid | int(10) | FK | Id of organization |
| 3 | etypeid | int100) | FK | Type of requested event |
| 4 | reventname | varchar(100) |  | Name of requested event |
| 5 | rstatus | varchar(50) |  | Status of request:  pending/approved/rejected |

### tbl\_payment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.NO** | **Field** | **Type** | **Key** | **Description** |
| 1 | pid | int(10) | PRI | Id of payment |
| 2 | vid | int(10) | FK | Id of volunteer |
| 3 | eid | int(10) | FK | Id of event |
| 4 | pdate | date |  | Date of payment |
| 5 | pamt | int(20) |  | Payment amount |
| 6 | pstatus | varchar(50) |  | Status of payment:pending/paid/donation |

### tbl\_volunteerevents

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.NO** | **Field** | **Type** | **Key** | **Description** |
| 1 | veid | int(10) | PRI | Id |
| 2 | vid | int(10) | FK | Id of volunteer |
| 3 | etypeid | int(10) | FK | Id of event type interested to the volunteer |

### tbl\_activity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.NO** | **Field** | **Type** | **Key** | **Description** |
| 1 | ayid | int(10) | PRI | Id of activity |
| 2 | vid | int(10) | FK | Id of volunteer |
| 3 | eid | int(10) | FK | Id of event |
| 4 | aystatus | varchar(50) |  | Activity status of volunteer:  complete/incomplete |

# TESTING

### Test Cases

### Test Case 1: Check results on entering valid User ID & Password

### Description: This test case verifies the login functionality when valid user credentials are entered.

### Inputs:

### User ID: valid email address

### Password: valid password associated with the user ID

### Expected Output: The user should be successfully logged into the system.

### Pass/Fail Criteria: The user should be able to access the system features and the login attempt should be considered successful.

### Test Case 2: Check results on entering invalid Email-id & Password

### Description: This test case checks how the system handles invalid user credentials.

### Inputs:

### User ID: invalid email address

### Password: invalid password

### Expected Output: The login attempt should be rejected, and the user should not be granted access to the system.

### Pass/Fail Criteria: The system should block the unauthorized login attempt and display an error message indicating invalid credentials.

### Test Case 3: Check results when User ID is empty and login button is pressed

### Description: This test case verifies the behavior of the system when the user ID field is left empty.

### Inputs:

### User ID: empty

### Password: valid or invalid password

### Expected Output: The login attempt should be rejected, and the user should not be granted access to the system.

### Pass/Fail Criteria: The system should block the unauthorized login attempt and display an error message indicating that the user ID field is required.

### Test Report

### Login Functionality

### Test Case 1: Check results on entering valid User ID & Password

### 

### Actual Result: The user was successfully logged into the system.

### Expected Result: The user should be successfully logged into the system.

### Pass/Fail: Pass

### Test Case 2: Check results on entering invalid Email-id & Password

### 

### Actual Result: The login attempt was rejected, and an error message indicating invalid credentials was displayed.

### Expected Result: The login attempt should be rejected, and the user should not be granted access to the system.

### Pass/Fail: Pass

### 

### Test Case 3: Check results when User ID is empty and login button is pressed

### 

### Actual Result: The login attempt was rejected, and an error message indicating that the user ID field is required was displayed.

### Expected Result: The login attempt should be rejected, and the user should not be granted access to the system.

### Pass/Fail: Pass

### Quality Measure:

### Total Passed Test Cases: 3

### Total Test Cases: 3

### Percentage of Passed Test Cases: 100%

### All the test cases for the login functionality have passed, indicating that the unauthorized access is being properly prevented. The login feature is functioning as expected, and the quality measure shows a 100% pass rate for the test cases.

### Sample Code used for testing

class LoginTest extends TestCase

{

public function testValidCredentials()

{

$loginService = new LoginService();

$vemail = sana.com';

$vpswd = 'sana';

$result = $loginService->login($vemail, $vpswd);

$this->assertTrue($result);

}

public function testInvalidCredentials()

{

$loginService = new LoginService();

$vemail = 'invalid@example.com';

$vpswd = 'invalidpassword';

$result = $loginService->login($vemail, $vpswd);

$this->assertFalse($result);

}

public function testEmptyFields()

{

$loginService = new LoginService();

$vemail = '';

$vpswd = 'password123';

$result = $loginService->login($vemail, $vpswd);

$this->assertFalse($result);

}

}

# TRANSITION

### SYSTEM IMPLEMENTATION

System Implementation is a crucial phase in the life cycle of the project, involving the deployment and utilization of the developed system. It encompasses various activities such as equipment setup, user training, and data migration. Similar to the example provided, the implementation process for the "Volunteer Management for Non-Profits and Community Groups" project involves careful planning and consideration of system requirements and constraints.

During the implementation phase, the system will be installed and put into operation. This can range from the conversion of existing applications to the complete replacement of computer systems. The implementation strategy and timeline will be determined at the outset of the project. Thorough testing of the system will be conducted to ensure its reliability and functionality.

Furthermore, user training will be an essential component of the implementation process. The organizations and volunteers will be provided with the necessary training to familiarize themselves with the system's features and functionalities. This training will enable them to effectively manage events, volunteering activities, and other related tasks within the system.

Additionally, the implementation phase will involve evaluating the changeover methods. The project team will carefully assess the effectiveness and efficiency of the chosen methods for transitioning from the old system to the new one. This evaluation will ensure a smooth and seamless integration of the new system into the existing organizational processes.

Once the system is fully implemented, it will be tailored to meet the specific requirements of the organizations and community groups using it. This may involve customizations and adjustments to align the system with their unique needs and workflows.

Overall, a well-executed implementation process is crucial to the successful deployment of the system and meeting the objectives of the project. It ensures that the new system is properly installed, users are trained, and the organization's requirements are met. While implementation itself does not guarantee organizational improvement, it serves as a vital step in preventing improper installation and lays the foundation for efficient system utilization..

### SYSTEM MAINTENANCE

System maintenance is a critical aspect of the "Volunteer Management for Non-Profits and Community Groups" project, ensuring the system's ongoing functionality, adaptability, and user satisfaction. The maintenance phase plays a vital role in addressing necessary changes, resolving errors, and enhancing the system to meet evolving requirements. This report highlights the key components of system maintenance and its significance in the project.

During system maintenance, the analyst undertakes several essential tasks:

1. Resolving necessary changes: The analyst promptly addresses user requests for modifications, incorporating updates and enhancements to align the system with the changing needs of non-profit organizations and community groups.
2. Correcting errors: Any identified errors or bugs in the system are meticulously examined and corrected to maintain a seamless user experience and ensure accurate functionality.
3. Enhancing or modifying the system: As new requirements and opportunities arise, the system is continuously enhanced and modified to optimize performance, usability, and efficiency. This may involve adding new features or improving existing ones to better serve the users' needs.
4. Assigning staff to perform maintenance activities: Competent staff members are assigned responsibilities for system maintenance tasks, ensuring dedicated resources and expertise are available to promptly address any issues that arise.
5. Providing for scheduled maintenance: Regular maintenance schedules are established to proactively manage system health, conduct routine checks, and perform necessary updates. This helps prevent potential problems, minimize downtime, and ensure the system's reliability.

System maintenance can be categorized into three main areas:

Corrective maintenance: This type of maintenance focuses on resolving residual errors present in the system upon its initial delivery, as well as errors that may have been introduced during subsequent maintenance activities. Corrective maintenance accounts for approximately 20% of the overall maintenance cost.

Adaptive maintenance: Adaptive maintenance involves adjusting the system to accommodate changes in the environment, such as updates to hardware, operating systems, or database systems. Approximately 20% of the maintenance cost is dedicated to adaptive maintenance.

Preventive maintenance: Preventive maintenance aims to proactively improve the software by enhancing its functionality and performance. This category accounts for over 50% of the maintenance costs, and it encompasses modifications to existing features, the addition of new functions, and overall system optimization.

Efficient system maintenance is crucial for the longevity and success of the "Volunteer Management for Non-Profits and Community Groups" project. By dedicating attention to maintenance tasks, the project team ensures that user requirements are continuously met, errors are promptly addressed, and the system remains reliable, adaptable, and responsive to evolving needs. This commitment to system maintenance aligns with the goal of delivering a high-quality solution and fostering long-term satisfaction for organizations, volunteers, and administrators alike..

### USER/OPERATIONAL MANUAL

### Operation Details:

### To operate the Volunteer Management system effectively, follow these instructions:

### Starting the system: [Provide instructions on how to start the system, including any necessary steps or configurations.]

### Shutting down the system: [Provide instructions on how to safely shut down the system and exit.]

### User Roles:

### The Volunteer Management system includes three user roles: Admin, Organizations, and Volunteers. Each role has specific responsibilities and privileges. The following are the key responsibilities for each role:

### Admin: Responsible for system management, including approving applications, managing entities, generating reports, and handling event requests.

### Organizations: Responsible for managing events, volunteer activities, payments, and requesting new events.

### Volunteers: Responsible for viewing and editing personal details, applying to volunteer, managing interests, viewing event preferences, and downloading certificates.

### Periodic Data Cleaning:

### To maintain data integrity, it is important to perform periodic data cleaning tasks. Please follow these guidelines:

### [Specify any specific data cleaning tasks, such as removing expired events or archiving inactive volunteers. Provide instructions on how to perform these tasks.]

### Configuration Management:

### To ensure system stability and data backup, consider the following:

### Backup frequency: [Specify the recommended frequency for backing up system-related files, such as configuration files or databases.]

### Backup procedure: [Provide instructions on how to perform backups, including any tools or scripts provided for automated backup.]

# ANNEXURE

### Document Glossary

### HTML - Hypertext Markup Language

### CSS - Cascading Style Sheets

### PHP - Hypertext Preprocessor

### XAMPP - Cross-Platform (X), Apache (A), MariaDB (M), PHP (P), Perl (P)

### API - Application Programming Interface

### UI - User Interface

### UX - User Experience

### SQL - Structured Query Language

### References:

#### Websites:

* 1. [https://getbootstrap.com/docs/4.5/getting-](https://getbootstrap.com/docs/4.5/getting-started/introduction/) [started/introduction/](https://getbootstrap.com/docs/4.5/getting-started/introduction/)
  2. <https://www.w3schools.com/>
  3. <https://www.youtube.com/>
  4. <https://www.freecodecamp.org/>

### User Interview Questionnaires:

1. How would you approach the system?
2. What about usability of this system?
3. What are normal project requirements

### Sample Project code:

**PHP code for approving or rejecting organization applications.**

if ($value == "approved") { foreach ($extract\_id as $key => $id) {

echo $sql = "UPDATE tbl\_organization SET ostatus = 'approved' WHERE oid = $id";

$db->query($sql);

}

}

else if($value=="rejected")

{

foreach($extract\_id as $key=>$id)

{

$sql = "Update tbl\_organization set ostatus='rejected' where oid=$id";

$db->query($sql);

}

}

**PHP code for viewing details of events.**

$prev\_date = trim($\_POST['estart']);

$next\_date = trim($\_POST['eend']);

$sql1 = "SELECT \* FROM tbl\_event WHERE estart BETWEEN '$prev\_date' AND '$next\_date'";

$result1 = $db->query($sql1);

if ($result1->num\_rows > 0) {

$serial=1;

?>

<table class="table table-bordered" id="dataTable" width="100%" cellspacing="0">

<tbody>

<tr>

<th style="text-align:center;"><b>S.No</b></th>

<th style="text-align:center;"><b>Event</b></th>

<th style="text-align:center;"><b>Time</b></th>

<th style="text-align:center;"><b>Location</b></th>

<th style="text-align:center;"><b>Required volunteers</b></th>

<th style="text-align:center;"><b>Applied volunteers</b></th>

<th style="text-align:center;"><b>Wage for volunteer</b></th>

<th style="text-align:center;"><b>Status</b></th>

</tr>

<?php

while($row = $result1->fetch\_assoc())

{

?>

<tr>

</form>

<td><?php echo $serial; ?></td>

<td style="justify-content:center; text-align:center" width="70%"> <?php echo $row["edescription"];?></td>

<td style="justify-content:center; text-align:center" width="70%"> <?php echo $row["etime"];?></td>

<td style="justify-content:center; text-align:center" width="70%"> <?php echo $row["elocation"];?></td>

<td style="justify-content:center; text-align:center" width="70%"> <?php echo $row["ereqno"];?></td>

<td style="justify-content:center; text-align:center" width="70%"> <?php echo $row["eappliedno"];?></td>

<td style="justify-content:center; text-align:center" width="70%"> <?php echo $row["ewage"]; echo "Rs";$serial=$serial+1;?></td>

<td style="justify-content:center; text-align:center" width="10%">

<?php

if($row["estatus"]=='complete') {?>

<span style="color:green" class="glyphicon glyphicon-edit fas fa-md mt-1 py-2 fa-check " aria-hidden="true"></span>

<?php

}

else if($row["estatus"]=='incomplete') { ?>

<span style="color:maroon" class="glyphicon glyphicon-edit fas fa-lg mt-1 py-2 fa-times" aria-hidden="true"></span>

<?php } ?>

</td>

</tr>

<?php

}

}}

?>

</tbody>

</table>

**PHP code for adding new events.**

$oid=$\_SESSION["oid"];

$elid = $\_POST['eventlist'];

$edescription = $\_POST['edescription'];

$estart = $\_POST['estart'];

$eend = $\_POST['eend'];

$etime = $\_POST['etime'];

$elocation = $\_POST['elocation'];

$eagell = $\_POST['eagell'];

$eageul = $\_POST['eageul'];

$ereqno = $\_POST['ereqno'];

$ewage = $\_POST['ewage'];

$sql = "insert into tbl\_event(elid,oid,edescription,estart,eend,etime,elocation,eagell,eageul,ereqno,ewage) values ($elid,$oid,'$edescription','$estart','$eend','$etime','$elocation',$eagell,$eageul,$ereqno,$ewage)";

if($db->query($sql) == TRUE)

{

$\_SESSION['msg'] = "Event added successfully !";

header("location: events.php");

}

**PHP code for managing events and volunteering activities.**

<tr>

<!-- <th>

<b><input type='button' id="delete" value='Delete' name='delete'></b></th> -->

<th style="justify-content:center; text-align:center;" >

<input style='vertical-align:bottom;width:14px;height:15px;' name='checkbox-main' type='checkbox' value='Select All' id='checkbox-main' onclick=selectAll() >

</th>

<th style="text-align:center;"><b>Event</b></th>

<th style="text-align:center;"><b>Date</b></th>

<th style="text-align:center;"><b>View</b></th>

<th style="text-align:center;"><b>Edit</b></th>

<th style="text-align:center;"><b>Applications</b></th>

<th style="text-align:center;"><b>Activities</b></th>

<th style="text-align:center;"><b>Delete</b></th>

<th style="text-align:center;"><b>Status</b></th>

</tr>

<?php

while($row = $result1->fetch\_assoc())

{

?>

<tr>

<td style="justify-content:center; text-align:center" width="5%">

<input type="checkbox" style="vertical-align:bottom;width:14px;height:15px" class="checkbox-child" name="check\_list[]" value=<?php echo $row['eid']; ?> >

</td>

</form>

<td style="justify-content:center; text-align:center" width="70%"> <?php echo $row["edescription"]; ?> </td>

<td style="justify-content:center; text-align:center" width="70%"> <?php echo $row["estart"]; ?> </td>

<td style="justify-content:center; text-align:center" width="10%">

<a href="view.php?eid=<?php echo $row['eid']?>" style="background-color:white;" class="btn btn-light a-btn-slide-text">

<span style="color:gray" class="glyphicon glyphicon-edit fas fa-info" aria-hidden="true"></span>

</a>

</td>

<td style="justify-content:center; text-align:center" width="10%">

<a href="editevent.php?eid=<?php echo $row['eid']?>" style="background-color:white;" class="btn btn-light a-btn-slide-text">

<span style="color:orange" class="glyphicon glyphicon-edit fas fa-pencil-alt" aria-hidden="true"></span>

</a>

</td>

<td style="justify-content:center; text-align:center" width="10%">

<a href="applications.php?eid=<?php echo $row['eid']?>" style="background-color:white;" class="btn btn-light a-btn-slide-text">

<!-- <a href="applicationtemplate.php?eid=<?php echo $row['eid']?>" style="background-color:white;" class="btn btn-light a-btn-slide-text"> -->

<span style="color:blue" class="glyphicon glyphicon-list fas fa-users" aria-hidden="true"></span>

</a>

</td>

<td style="justify-content:center; text-align:center" width="10%">

<a href="activity.php?eid=<?php echo $row['eid']?>" style="background-color:white;" class="btn btn-light a-btn-slide-text">

<span style="color:green" class="glyphicon glyphicon-tasks fas fa-tasks" aria-hidden="true"></span>

</a>

</td>

<td style="justify-content:center; text-align:center" width="10%">

<a href="delete.php?id=<?php echo $row['eid']; ?>" onclick="return confirm('Are you sure you want to delete?')"

style="background-color:white;" class="btn btn-light a-btn-slide-text">

<span style="color:red" class="glyphicon glyphicon-edit fas fa-md fa-trash" aria-hidden="true"></span>

</a>

</td>

<td style="justify-content:center; text-align:center" width="10%">

<?php

if($row["estatus"]=='complete') {?>

<span style="color:green" class="glyphicon glyphicon-edit fas fa-md mt-1 py-2 fa-check " aria-hidden="true"></span>

<?php

}

else if($row["estatus"]=='incomplete') { ?>

<span style="color:maroon" class="glyphicon glyphicon-edit fas fa-lg mt-1 py-2 fa-times" aria-hidden="true"></span>

<?php } ?>

</td>

</tr>

<?php

}

}?>

**PHP code for viewing events based on preferences and age.**

$sql1 = "SELECT \*

FROM tbl\_event

WHERE elid IN (

SELECT elid

FROM tbl\_eventlist

WHERE etypeid IN (

SELECT etypeid

FROM tbl\_volunteerevents

WHERE vid = $vid

)

)

AND ereqno > eappliedno

AND ($currentYear - (SELECT vage FROM tbl\_volunteer WHERE vid = $vid)) BETWEEN eagell AND eageul

AND NOT EXISTS (

SELECT \*

FROM tbl\_activity

WHERE tbl\_activity.eid = tbl\_event.eid

AND tbl\_activity.vid = $vid

)

AND estatus = 'incomplete'

AND eend > CURDATE()";

$result1 = $db->query($sql1);

**PHP code for applying to volunteer for an event.**

$eappliednonew=$eappliedno+1;

$vid=$\_SESSION["vid"];

$eid=$\_SESSION['eid'];

$ewage=$\_SESSION['ewage'];

$pstatus= $\_POST['pstatus'];

$date = date("Y-m-d");

$sql = "INSERT INTO tbl\_payment (vid, eid, pdate, pamt, pstatus) VALUES ($vid, $eid, '$date', $ewage, '$pstatus')";

$db->query($sql);

$sql17="UPDATE tbl\_event SET eappliedno = $eappliednonew WHERE eid = $eid";

if ($db->query($sql17) === TRUE) {

echo " ";

} else {

echo "Error updating record: " . $db->error;

}

$\_SESSION['eappliedno'] = $eappliednonew;

$sql12 = "insert into tbl\_activity(vid,eid,aystatus) values ($vid,$eid,'incomplete')";

if($db->query($sql12) == TRUE)

{

$\_SESSION['msg'] = "Your application is submitted ";

header("location: events.php");

}

if ($pstatus == "donation") {

$obalancenew = $obalance + $ewage;

$sql21 = "UPDATE tbl\_organization SET obalance = $obalancenew WHERE oid = $oid";

$db->query($sql21); }