Software Engineering Tool

and Practices Project

Tittle: Charity Website

Group Members Id

Firdos Mohammed…………………………3927/13

Marta Dame……………………………………1656/13

Abdulmajid Tsegaye………………………..0056/13

Esmael Eshetu…………………………………0932/13

Zelalem Beyene……………………………….4606/13

Bahrean Assefa………………………………..4605/13

Contents

[Functional requirements 2](#_Toc140246311)

[Non-functional requirements 3](#_Toc140246312)

[Use case Diagram 4](#_Toc140246313)

[Activity Diagram 5](#_Toc140246314)

[Make Donation Standard UML Activity Diagram 5](#_Toc140246315)

[Sign up for Volunteer Activity Standard UML Activity Diagram 6](#_Toc140246316)

[View Project Listings Classic UML Activity Diagram 7](#_Toc140246317)

[Sequence Diagram 8](#_Toc140246318)

[Make Donation Sequential Diagram 8](#_Toc140246319)

[View Project Listings Sequential Diagram 9](#_Toc140246320)

[Sign up for Volunteer Activity Sequential Diagram 10](#_Toc140246321)

[Class Diagram and source code 11](#_Toc140246322)

[Class Diagram 11](#_Toc140246323)

[Source Code 12](#_Toc140246324)

[Component Diagram 16](#_Toc140246325)

[Deployment Diagram 17](#_Toc140246326)

# 

# Functional requirements

User Management:

* Allow users to register an account on the website.
* Provide user login functionality for authenticated access to the website
* Enable users to update their profile information and manage their account settings.

Donation Submission:

* Enable users to make donations through the website.
* Provide a user interface to select donation options, including predefined amounts or custom amounts
* Process donation transactions securely and provide confirmation to the user.
* Store donation details, including the donor's information and the donation amount.

Volunteer Signup:

* Allow users to sign up as volunteers for various activities.
* Provide a user interface to view available volunteer activities and their details.
* Enable users to select and sign up for specific activities
* Store volunteer details, including the user's information and the selected activity.

Activity Management:

* Enable administrators to create, update, and delete volunteer activities
* Provide an interface for administrators to manage activity details, including the activity name, description, date, time, and location.
* Display the list of activities and their details to users

# Non-functional requirements

Performance:

* The website should have fast page load times to ensure a seamless user experience.
* Response times for critical operations, such as donation submission and volunteer signup, should be minimal.
* The system should be able to handle a high volume of concurrent users without significant performance degradation.

Security:

* User data, including personal information and payment details, should be securely stored and transmitted.
* Implement appropriate encryption techniques to protect sensitive data.
* Ensure secure authentication and authorization mechanisms to prevent unauthorized access.
* Regularly update security measures to address emerging threats and vulnerabilities.

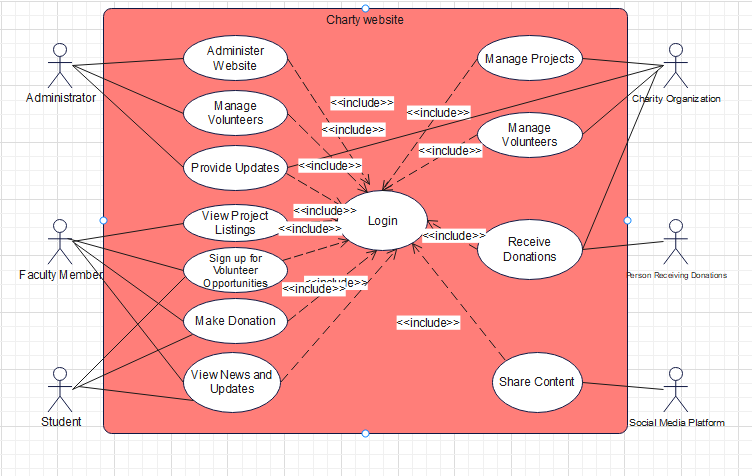
Reliability and Availability:

* The website should be highly available and accessible to users without frequent downtime.
* Implement backup and disaster recovery mechanisms to ensure data integrity and system availability in case of failures or disasters.
* Monitor the system proactively to identify and address any potential issues or performance bottlenecks.

Usability and User Experience:

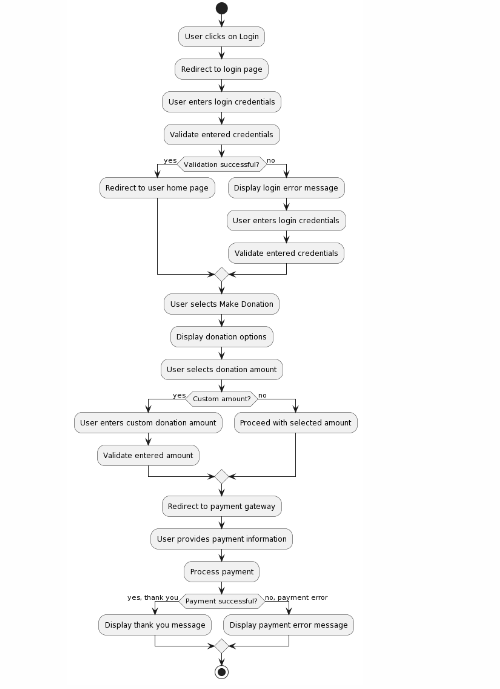
* The website should have an intuitive and user-friendly interface.
* Ensure clear navigation and ease of use for users of varying technical expertise.
* Provide informative and helpful error messages to guide users in case of any issues.

# Use case Diagram

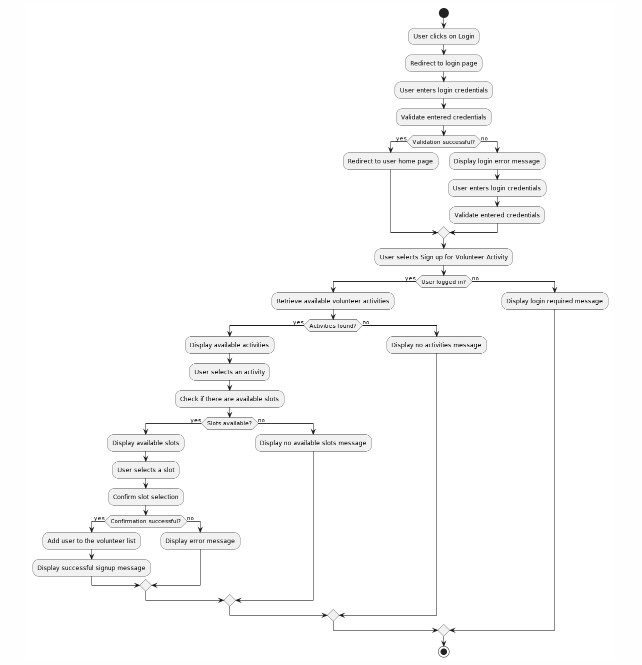


# Activity Diagram

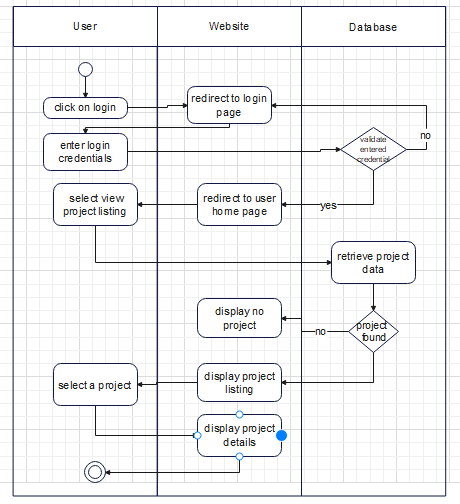
Make Donation Standard UML Activity Diagram:



Sign up for Volunteer Activity Standard UML Activity Diagram:

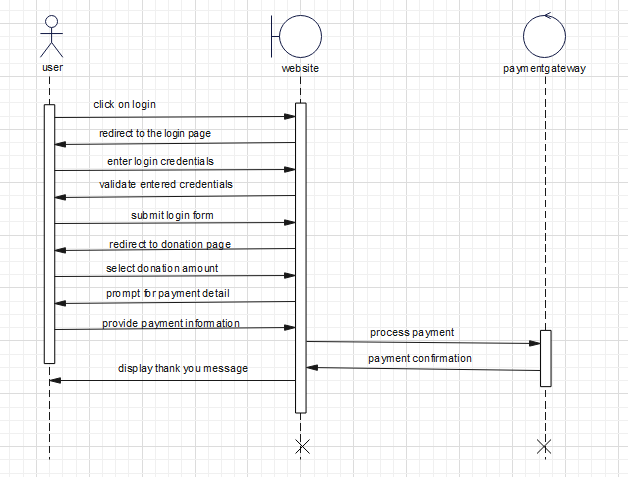


View Project Listings Classic UML Activity Diagram:

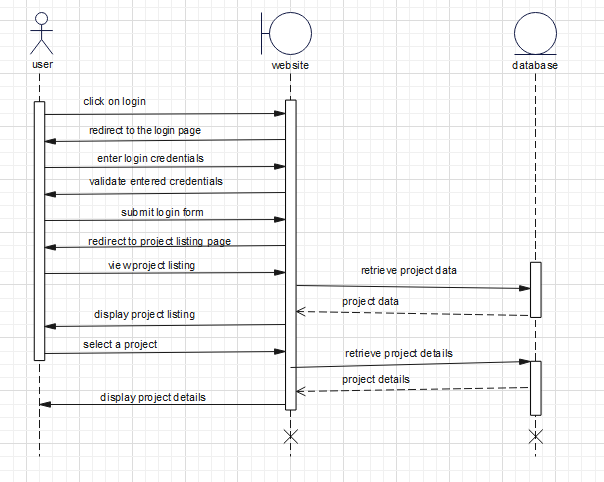


# Sequence Diagram

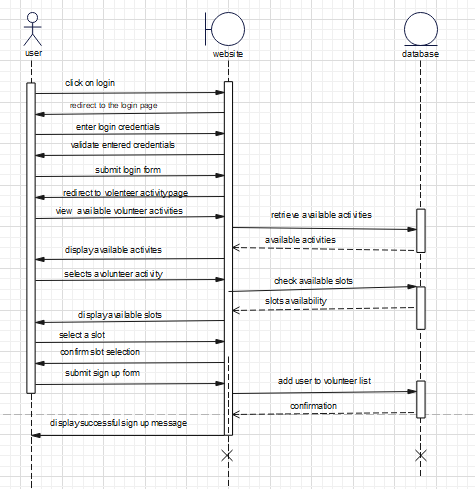
Make Donation Sequential Diagram:



View Project Listings Sequential Diagram:

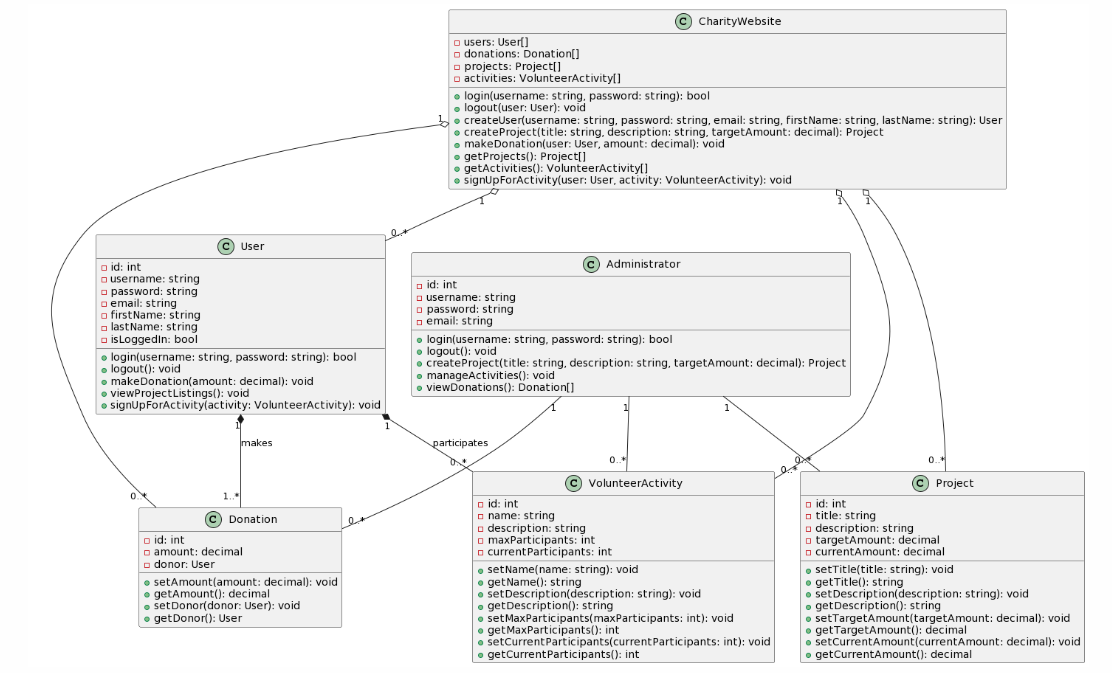


Sign up for Volunteer Activity Sequential Diagram:



# Class Diagram and source code

## Class Diagram



Source Code with php

class User {

-id: int

-username: string

-password: string

-email: string

-firstName: string

-lastName: string

-isLoggedIn: bool

+login(username: string, password: string): bool

+logout(): void

+makeDonation(amount: decimal): void

+viewProjectListings(): void

+signUpForActivity(activity: VolunteerActivity): void

}

class Donation {

-id: int

-amount: decimal

-donor: User

+setAmount(amount: decimal): void

+getAmount(): decimal

+setDonor(donor: User): void

+getDonor(): User

}

class Project {

-id: int

-title: string

-description: string

-targetAmount: decimal

-currentAmount: decimal

+setTitle(title: string): void

+getTitle(): string

+setDescription(description: string): void

+getDescription(): string

+setTargetAmount(targetAmount: decimal): void

+getTargetAmount(): decimal

+setCurrentAmount(currentAmount: decimal): void

+getCurrentAmount(): decimal

}

class VolunteerActivity {

-id: int

-name: string

-description: string

-maxParticipants: int

-currentParticipants: int

+setName(name: string): void

+getName(): string

+setDescription(description: string): void

+getDescription(): string

+setMaxParticipants(maxParticipants: int): void

+getMaxParticipants(): int

+setCurrentParticipants(currentParticipants: int): void

+getCurrentParticipants(): int

}

class CharityWebsite {

-users: User[]

-donations: Donation[]

-projects: Project[]

-activities: VolunteerActivity[]

+login(username: string, password: string): bool

+logout(user: User): void

+createUser(username: string, password: string, email: string, firstName: string, lastName: string): User

+createProject(title: string, description: string, targetAmount: decimal): Project

+makeDonation(user: User, amount: decimal): void

+getProjects(): Project[]

+getActivities(): VolunteerActivity[]

+signUpForActivity(user: User, activity: VolunteerActivity): void

}

class Administrator {

-id: int

-username: string

-password: string

-email: string

+login(username: string, password: string): bool

+logout(): void

+createProject(title: string, description: string, targetAmount: decimal): Project

+manageActivities(): void

+viewDonations(): Donation[]

}

User "1" \*-- "1..\*" Donation : makes

User "1" \*-- "0..\*" VolunteerActivity : participates

CharityWebsite "1" o-- "0..\*" User

CharityWebsite "1" o-- "0..\*" Donation

CharityWebsite "1" o-- "0..\*" Project

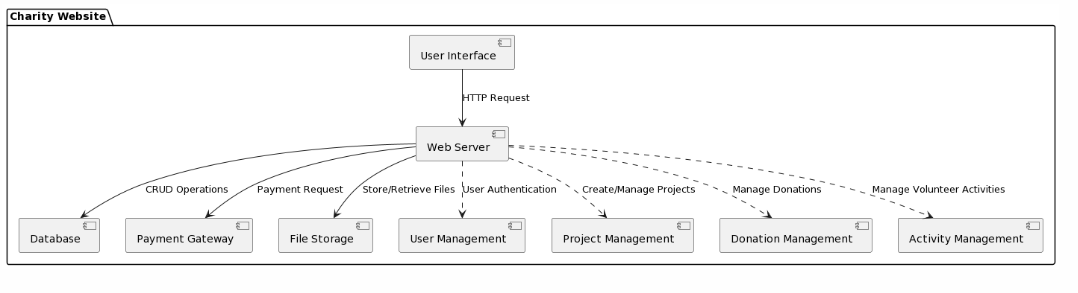
CharityWebsite "1" o-- "0..\*" VolunteerActivity

Administrator "1" -- "0..\*" Project

Administrator "1" -- "0..\*" VolunteerActivity

Administrator "1" -- "0..\*" Donation

# Component Diagram



# Deployment Diagram

