# Analysis Report

## Project scope:

The charity management system as a web application aims to streamline and automate various tasks related to charity management, including donations, fundraising, volunteer management, event management, etc. The project scope includes developing a user-friendly interface and implementing functionalities to support the above-mentioned tasks.

## Functional requirements:

1) Donor management

2) Fundraising management

3) Volunteer management

4) Event management

5) Reporting and analytics

6) Payment processing

7) User management

8) Email and SMS notifications

## Non-functional Requirements:

1) User-friendly interface

2) Secure login and authentication

3) Responsive design for mobile devices

4) Data protection and privacy

5) Scalability

6) Performance and reliability

7) Integration with external payment gateways

## User Interface Requirements:

1) Simple and intuitive design

2) Clear and concise navigation

3) User-friendly forms for data entry

4) Data visualization and reports

5) Mobile-responsive design

## Use Cases

### Level 0:

1) Donor Management

2) Fundraising Management

3) Volunteer Management

4) Event Management

5) Reporting and Analytics

6) Payment Processing

7) User Management

## Level 1:

1.1. Add/Edit/Delete Donors

1.2. View Donor History

2.1. Create Fundraising Campaign

2.2. Manage Donations

3.1. Recruit Volunteers

3.2. Manage Volunteer Schedules

4.1. Create and Manage Events

4.2. Track Attendance and Registrations

5.1. Generate Reports

5.2. Analyze Fundraising Trends

6.1. Process Donations

6.2. Manage Payment Transactions

7.1. Add/Edit/Delete Users

7.2. Assign Roles and Permissions

## Pre-conditions:

1) The user must be logged in

2) All required information must be entered

## Post-conditions:

1) Data is saved and updated in the database

2) Data is saved and updated in the database

3) Reports are generated and available for viewing.

## Business Rules:

1) Donors must provide contact information and payment details

2) Fundraising campaigns have a deadline and a target amount

3) Volunteer schedules must be approved by event organizers

4) Attendance at events must be confirmed

5) Reporting and analytics

6) User roles and permissions must be defined for security reasons.

## Assumptions:

1) The system will be used by staff and volunteers of the charity organization.

2) The system will be accessible from a web browser on desktop and mobile devices.

3) The system will integrate with external payment gateways for processing donations.

## Challenges:

1) Ensuring data security and privacy

2) Developing a user-friendly interface for a wide range of users

3) Integrating with external payment gateways

4) Ensuring scalability and performance under high traffic conditions.

## Use Cases:

A use case diagram is a visual representation of the system's functionalities, actors, and relationships between them.

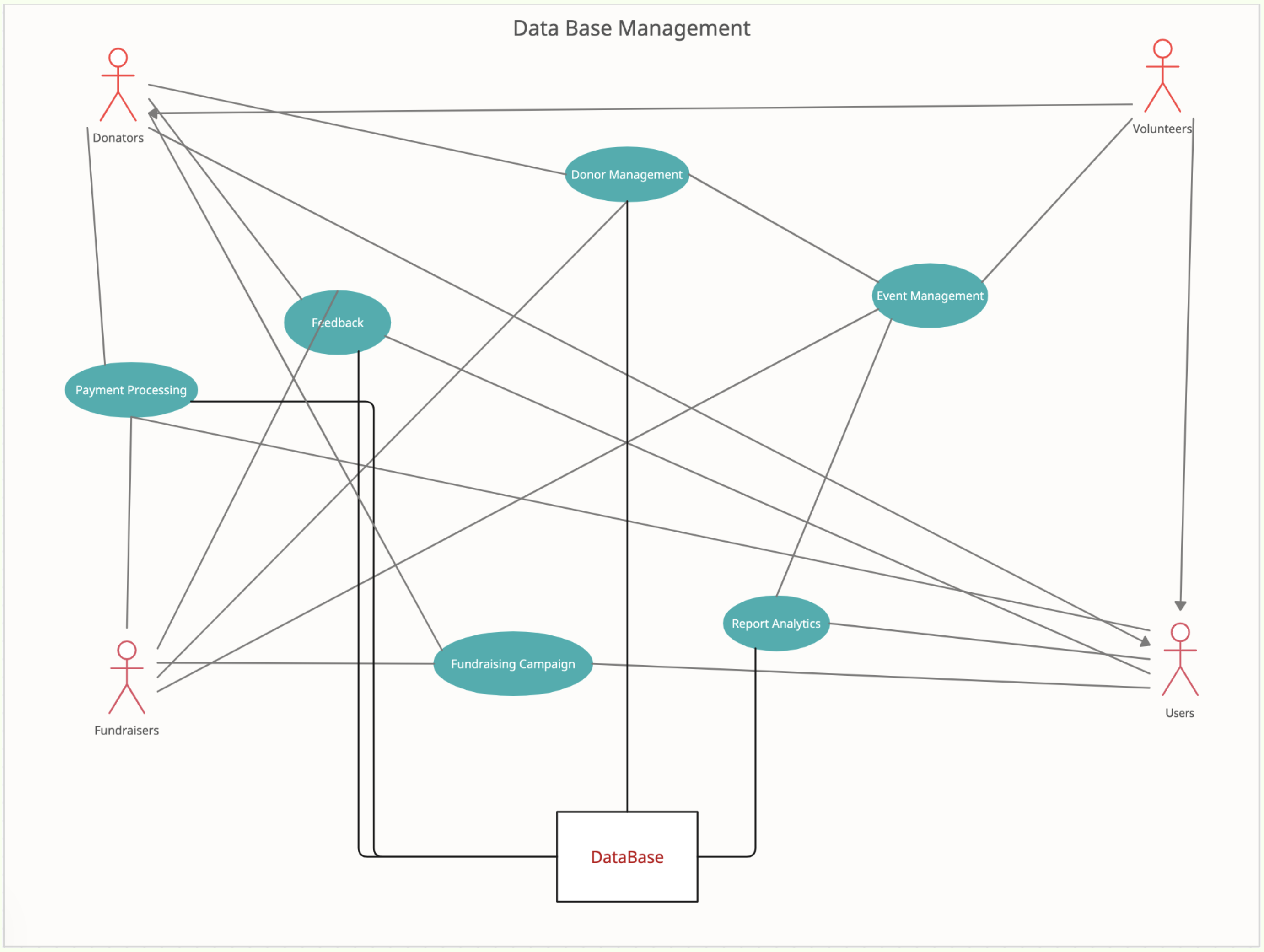
### Level 0 Use Case Diagram:

The main actors in the system are the Donors, Fundraisers, Volunteers, and Users.

The system will have several use cases, including Donor Management, Fundraising Management, Volunteer Management, Event Management, Reporting and Analytics, Payment Processing, and User Management.

Each use case represents a distinct functionality of the system and is shown as an oval shape.

The actors interact with the use cases, and this interaction is represented by a directed arrow connecting the actor and the use case.



## Level 1 Use Case Diagram:

The Level 1 use case diagram will provide more detail on each of the functionalities mentioned in the Level 0 diagram.

For example, the Donor Management use case in Level 0 will have several Level 1 use cases, such as Add/Edit/Delete Donors and View Donor History.

The Level 1 use cases will be connected to the Level 0 use cases they belong to and will be shown as smaller ovals within the larger oval representing the Level 0 use case.

The relationships between the actors and the Level 1 use cases will be similar to those in the Level 0 diagram, represented by directed arrows.

## Class Diagram:

### Identified Classes, attributes, and methods for the Level 1 use cases of the charity management system:

Classes: A class is a blueprint for creating objects. In object-oriented programming, classes represent real-world objects and their attributes and behaviors. For the charity management system, you can identify the following classes for the Level 1 use cases:

Donor: This class represents the donor and can have the following attributes:

Donor ID

Name

Contact Information (phone, email, address)

Payment Information (credit card, bank account)

Donation History

Fundraising Campaign: This class represents the fundraising campaign and can have the following attributes:

Campaign ID

Name

Description

Start and End Date

Target Amount

Donor Information

Donations Received

Volunteer: This class represents the volunteer and can have the following attributes:

Volunteer ID

Name

Contact Information (phone, email, address)

Skills and Availability

Volunteer History

Event: This class represents the event and can have the following attributes:

Event ID

Name

Description

Date and Time

Location

Attendance and Registrations

Report: This class represents the report and can have the following attributes:

Report ID

Name

Description

Date and Time

Data and Metrics

Payment: This class represents the payment and can have the following attributes:

Payment ID

Donor Information

Payment Amount

Payment Method

Payment Date and Time

User: This class represents the user of the system and can have the following attributes:

User ID

Name

Contact Information (phone, email, address)

Roles and Permissions

Login Information (username and password)

Methods: Methods are the behaviors or functions of the objects in a class. For the charity management system, you can identify the following methods for the classes mentioned above:

## Donor:

add Donor ()

Edit Donor ()

delete Donor ()

View Donor History ()

## Fundraising Campaign:

create Campaign ()

manage Donations ()

## Volunteer:

recruit Volunteer ()

Manage Volunteer Schedules ()

## Event:

create Event ()

manage Event ()

track Attendance ()

## Report:

generate Report ()

analyze Data ()

Payment:

process Payment ()

Manage Payment Transactions ()

## User:

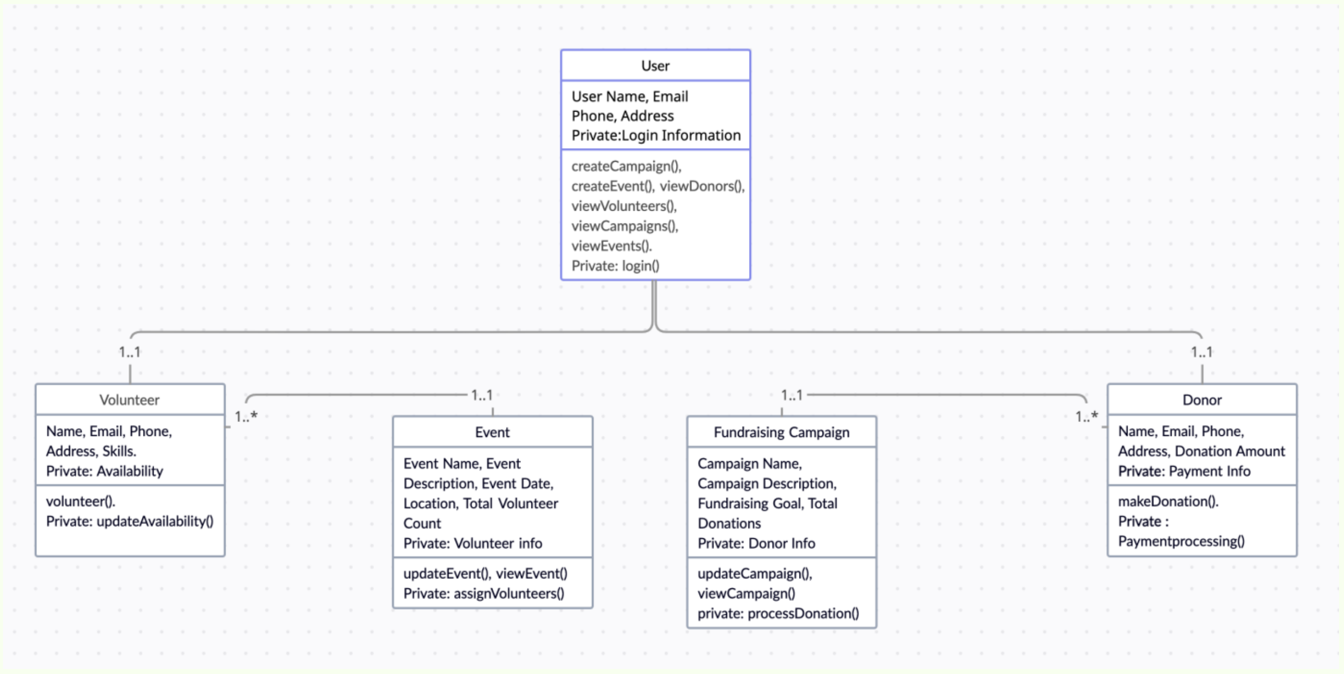
add User ()

Edit User ()

delete User ()

assign Roles ()

## Diagram:

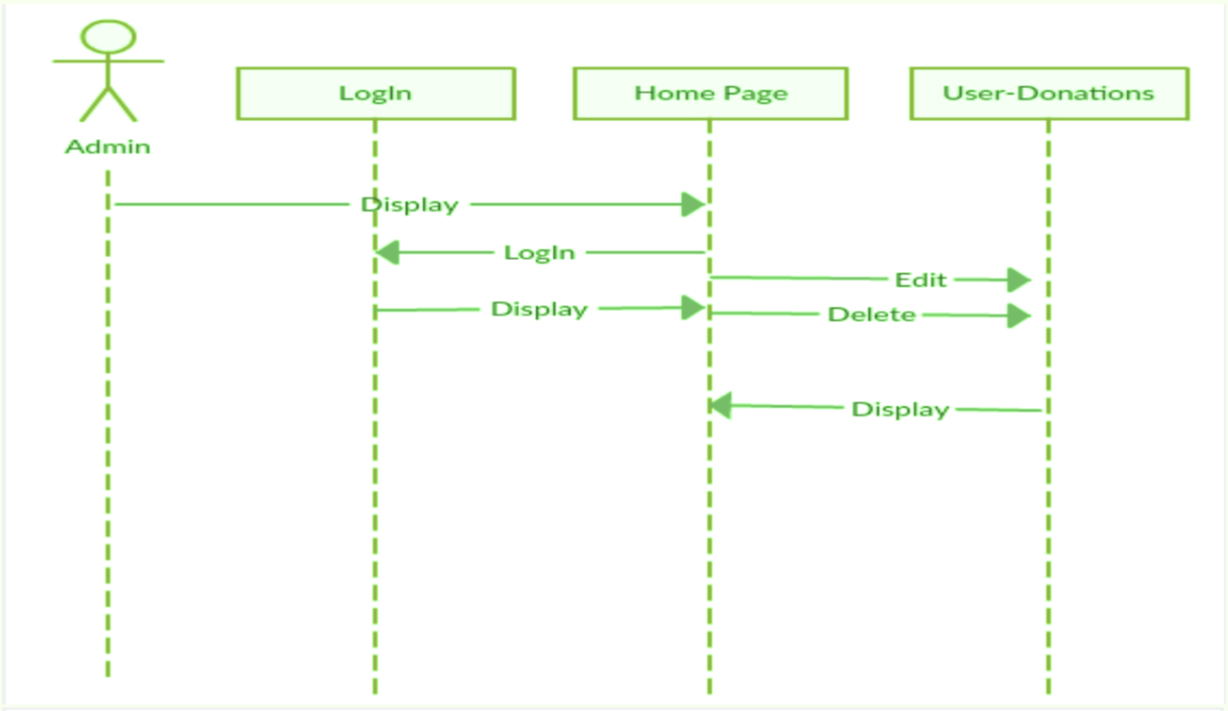


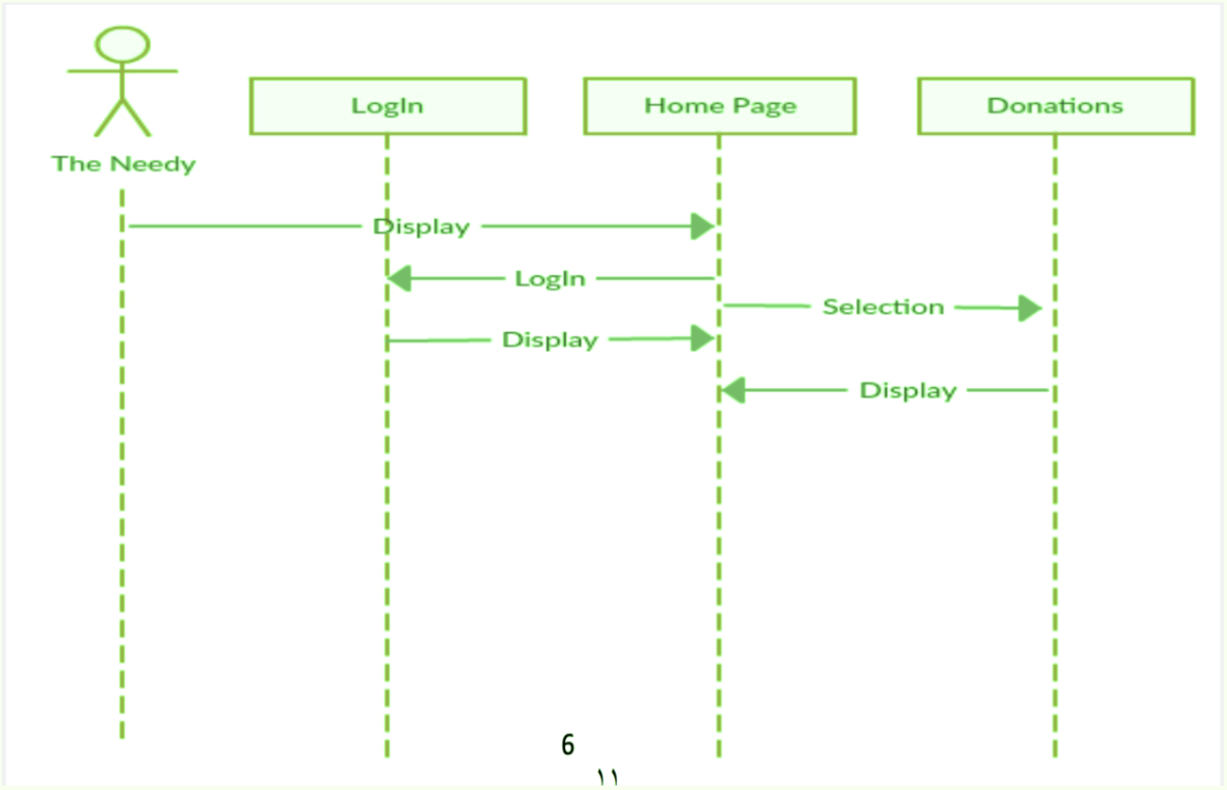
## Sequence Diagram:

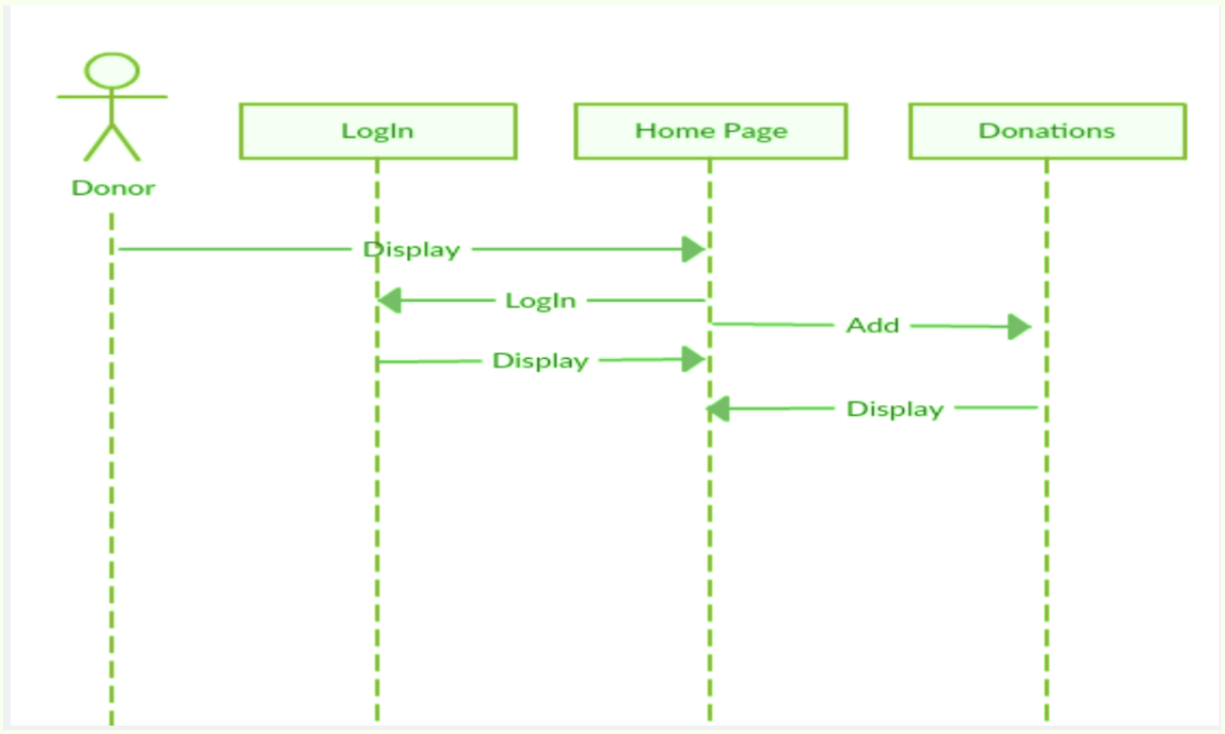
A sequence diagram is a visual representation of the interactions between objects in a system. It shows the flow of messages between objects over time.

The actors involved in the interactions could be the Donor, Fundraising Campaign, Event, Volunteer, and User. Each actor would be represented by a lifeline on the sequence diagram, showing the duration of their involvement in the interaction.

## Diagram:


Admin
LogIn
Home Page
User-Donations
Display
LogIn
Edit
Display
Delete
Display


The Needy
LogIn
Home Page
Donations
Display
LogIn
Selection
Display
Display



## Team name, Student names and Roll numbers

Team name: Solo Levelling

Student names: Pranay Karthik

Roll numbers: 21BCS084

## Glossary/References:

The whole report was written in intelligible language, I hope there won't be any necessary glossary.