# Introduction

This paper presents the technical documentation for the development of the donor database system for EDU Youth Foundation. It aims to manage donor information, events, and communication efficiently. The report describes the design, implementation, and functionality of the database, hence acting like a guide on how the system works in accomplishing the mission of the organization, which is quality education. The database has been designed to enhance organizational efficiency, facilitate access to data, and enable decision-making within the organization.

This document provides detailed information on how the database has been structured, what functionality it achieves, and how it meets the needs of the organization. This will include discussion on the system architecture, the tools and technologies adopted, the methodology followed in developing the system, and how certain features have been specifically implemented to address the organization's needs. This documentation serves as a reference not only to the current stakeholders but also for any future upgrade and scalability of the system. The intention of this report is to make the database system maximally usable and impactful by providing clear and structured insight into it.

# Database Requirements

## User requirements

* Donor:

The donor will be able to:

* Add their information to the system
* View his information
* View past donations, by date, amount, event, and campaign.
* To requested to delete his account in case they no longer want to participate.
* View events details
* Staff:

The employee can:

* Add new donors to the system
* Modify donors information
* Approve the donor account deletion requests
* View past donations, by date, amount, event, and campaign
* View events details
* Modify events details in case there is any changes

## System Requirements

* Software Requirements:
* **Operating System:** The operating system should be compatible with Windows, macOS, and Linux.
* **Database:** The system should be built using a relational database management system such as MySQL
* Hardware Requirements:
* **Server**: The system needs a web server to host the application.
* **Storage:** The system should have enough free disk space to store information about donors, logs of communications, event files, and records of donations made. This depends on the base size of donors.
* **Backup:** The system should have a reliable backup system for database and file backups to ensure data integrity and recovery.

## Data requirements

# Database Design

## Conceptual Design

**Definition**

Conceptual design is the abstract representation of a system or database that captures the very essence of the data and their interrelationships. It provides explicit concentration on the understanding and determination of the business needs and their translation into an effective logical structure. The conceptual design does not concern itself with the details of technical implementation but, in fact, finds the entities, their attributes, and relationships in a manner that could easily be comprehended by the stakeholders.

**Objective**

The aim of the conceptual design is to bridge the gap between business requirements and technical implementation. It aims to:

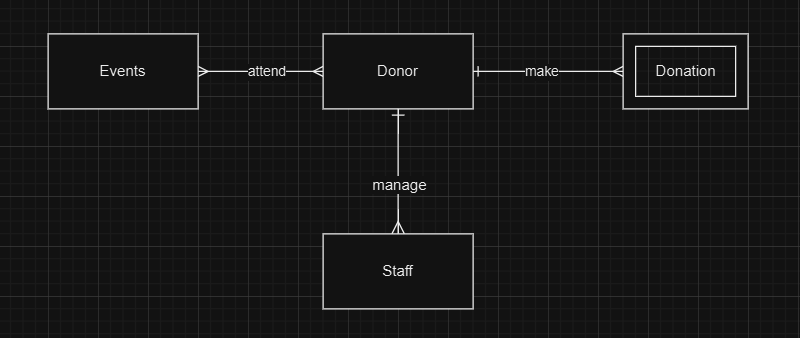
1. Clearly define the structure and scope of the data.
2. Identify key data elements and their relationships.
3. Ensure alignment with organizational objectives and user needs.
4. Provide a basis for subsequent logical and physical database designs.

**Components**

The major components of a conceptual design include:

* **Entities:** Represent the major objects or concepts in the system. In my case, donors, events, communication, and donations.
* **Attributes:** Properties or characteristics of the entities. For example, donor name, event date.
* **Relationships:** Links between entities. For example, donors sponsoring events, events having attendees.

## My Conceptual Design



The conceptual design above provides an overview of the entities and relationships among them in this database. For example, Donor and Event have M:N relationships, which show that one donor can attend multiple events, and at the same time, any event can have more than one attendee. It is represented by the attend connection. The entities Donor and Donation also have a 1:M relationship as one donor may make more than one donation. This is represented through the make connection. Donor and Staff entities also exist in a 1:M relationship as one staff member will manage all the donors. This is depicted through the manage connection. This conceptual design specifies the key entities and their interaction, thus providing a proper basis for logical and physical designs.

# Mapping

## My attributed with their types

* Donor (Donor ID, ssn, Donor name, Registration date, Email, phone number, Adress, Birth date)
* Staff (Staff ID, ssn, name, salary, level, Phone, Address, email, birth date)
* Event (Event ID, Event name, Event date, Event location)
* Communication (communication ID, Type, Date)
* Donation (Donation ID, Amount, Date)

|  |  |  |
| --- | --- | --- |
| Attribute | Type | Description |
| Entity1: Donor | | |
| Donor ID (PK) | Simple, Single-valued | Unique identifier for the donor |
| SSN | Simple, Single-valued | Social Security Number of the donor |
| Donor name | Composite | Consists of first name, middle name, and last name |
| Registration date | Simple, Single-valued | Date the donor registered in the system |
| Email | Simple, Single-valued | Email address of the donor |
| Phone number | Multivalued | A donor may have multiple phone numbers |
| Address | Composite | Includes street, city, state, and zip code |
| Birth date | Simple, Single-valued | Date of birth of the donor |
| Age | Derived |  |
| Entity2: Staff | | |
| Staff ID (PK) | Simple, Single-valued | Unique identifier for the staff member |
| Name | Composite | Consists of first name, middle name, and last name |
| Salary | Simple, Single-valued | Salary of the staff member |
| Level | Simple, Single-valued | Level or position of the staff member |
| Phone | Multivalued | A staff member may have multiple phone numbers |
| Address | Composite | Includes street, city, state, and zip code |
| Email | Simple, Single-valued | Email address of the staff member |
| Birth date | Simple, Single-valued | Date of birth of the staff member |
| Age | derived |  |
| Entity3: Event | | |
| Event ID (PK) | Simple, Single-valued | Unique identifier for the event |
| Event name | Simple, Single-valued | Name of the event |
| Event date | Simple, Single-valued | Date when the event will take place |
| Event location | Composite | Includes venue name, city, and zip code |
| Entity5: Donation | | |
| Donor ID (FG , Pk) | Simple, Single-valued | Unique identifier for the donation |
| Amount | Simple, Single-valued | Amount of the donation |
| Date | Simple, Single-valued | Date when the donation was made |

## Step1: Mapping of Regular (strong) Entity Types

In step 1, the strong entity should be mapped by resolving composite attributes and separating them into simpler attributes. Additionally, any multi-valued or complex attributes should be extracted and moved to a new, separate table. Finally I will delete the attributes derived.

* Donor (Donor ID, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, country, city, street name)
* Donors Phone (Donor ID, Phone Number)
* Staff (Staff ID, Staff First Name, Staff Last Name, Salary, Level, Email, Birth Date, country, city, street name)
* Staff Phone (Staff ID, Phone Number)
* Event (Event ID, Event Name, Event Date, Venue Name, City, street name)

## Step2: Mapping of Weak Entity Types

* Donation (Donor ID, Amount, Date)

## Step3: Mapping of 1:1 Relation types

In my scenario there is no 1:1 relation

## Step4: Mapping of 1:M Relation types

In this step we should make a link between tables by adding the PK of the (1) relation table to the many relation table as FK:

* Staff (Staff (Staff ID, Donor Id, Staff First Name, Staff Middle Name, Staff Last Name, Salary, Level, Email, Birth Date, country, city, street name)

## Step5: Mapping of M:N Relation types

In this step, we will represented as new table. And the primary key of this table will be combination of the primary keys of the two ends of the relation:

Participation (Donor ID, Event Id, participation date.

## Final schema

**Donor** (Donor ID, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name)

**Donors Phone** (Donor ID, Phone Number)

**Staff** (Staff ID, Staff First Name, Staff Last Name, Salary, Level, Email, Birth Date, Country, City, Street Name, Donor ID)

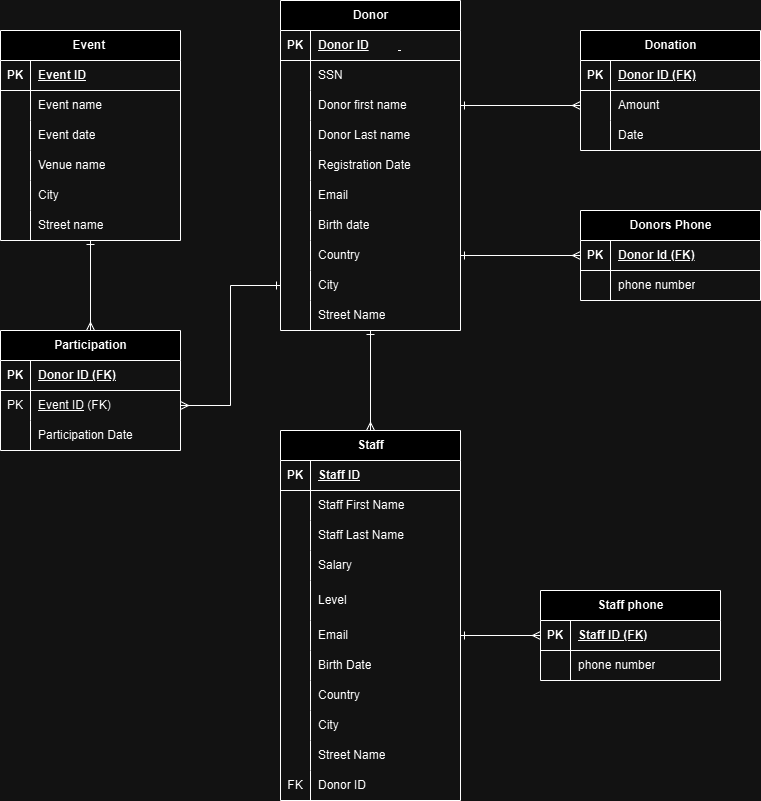
**Staff Phone** (Staff ID, Phone Number)

**Event** (Event ID, Event Name, Event Date, Venue Name, City, Street Name)

**Donation** (Donor ID, Amount, Date)

**Participation** (Donor ID, Event ID, Participation Date)

## Logical design

The logical design delineates how the conceptual database is organized into well-structured entities and relationships. The data should be stored in a normalized format, so redundancy is avoided and there is enhancement in consistency. It involves taking the identified entities, attributes, and their relationships from the conceptual model into relational tables. It will provide key elements such as primary keys to uniquely identify the records, foreign keys that link tables together, and associative tables that maintain many-to-many relationships. Below is the logical design, which shows how these things can be systematically put forward to represent the database structure in a clear and lucid fashion for the application:

Above is the logical design with all the attributes, keys, and the relationships that exist among the entities of the database. The relationship of Donor to Donors Phone would be 1:M-for one donor, there might be many phone numbers-while for Donor to Donation, the relationship would be 1:M too, since one donor can give multiple donations. The Participation entity resolves the many-to-many relationship between the entities Donor and Event; it contains a composite primary key, which includes two foreign keys: one coming from the Donor entity, namely, Donor ID, and another from the Event entity, namely, Event ID. This design ensures that all relationships are well-defined and that primary and foreign keys have been used in order to maintain data integrity and correctly represent the real-world links of these entities.

## Physical Design

## Normalization

### 1st NF

The First Normal Form ensures that the database schema adheres to the following principles:

1. Each cell in a table should contain a single, atomic value.
2. Each record in the table must be unique, with no duplicate rows.
3. Values stored in a column should belong to the same domain, ensuring data consistency.
4. All columns in a relation should have unique names to avoid ambiguity.

In this step, composite attributes are decomposed into atomic attributes, multi-valued attributes are extracted into separate relations, and any violations of 1NF are resolved. The table below identifies the violations in the schema and provides solutions for achieving 1NF compliance:

|  |  |  |  |
| --- | --- | --- | --- |
| Relations | Attributes | Violation description | Solution – Relations |
| Donor (Donor ID, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name) | Donor ID, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name | The Donor entity is in First Normal Form (1NF) as each cell contains atomic values, and there are no multi-valued attributes. Each record is uniquely identified by the Donor ID, ensuring uniqueness. All columns store values of the same domain, such as SSN being numeric and Birth Date being a date. Additionally, all column names are unique, with no ambiguity, meeting the 1NF requirements | **Donor** (Donor ID, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name) |
| Event (Event ID, Event Name, Event Date, Venue Name, City, Street Name) | Event ID, Event Name, Event Date, Venue Name, City, Street Name | Event ID, Event Name, Event Date, Venue Name, City, Street Name  Description: The following structure of the Event entity is already in 1NF since each cell contains atomic values and there are no multi-valued attributes. Each record has a unique identifier-the Event ID-assuring uniqueness. All columns are of the same domain, such as the Event Date being a date, while the Venue Name is text. All column names are unique; there's no ambiguity. Therefore, it meets the requirements for 1NF. | **Event** (Event ID, Event Name, Event Date, Venue Name, City, Street Name) |
| Donation (Donor ID, Amount, Date) | Donor ID, Amount, Date | First Normal Form says that each cell should contain an atomic value, meaning there are no multi-valued attributes. This is so because each record is uniquely identified by the composite key, Donor ID and Date. All columns store values of the same domain: Amount is numeric; Date is a date. All column names are unique; no ambiguity - this meets 1NF criteria. | **Donation** (Donor ID, Amount, Date) |
| Staff (Staff ID, Staff First Name, Staff Last Name, Salary, Level, Email, Birth Date, Country, City, Street Name, Donor ID) | Staff ID, Staff First Name, Staff Last Name, Salary, Level, Email, Birth Date, Country, City, Street Name, Donor ID | Staff ID, Staff First Name, Staff Last Name, Salary, Level, Email, Birth Date, Country, City, Street Name, Donor ID  Description: The Staff entity is in 1NF, and in each cell, the value is atomic, with no multi-valued attribute; each record is uniquely identified and is represented by the staff ID. Each column is of a consistent type or domain-the value of Salary numeric and Birth Date is a date-and every column is named uniquely without any possible ambiguity, meeting the requirements for 1NF. | **Staff** (Staff ID, Staff First Name, Staff Last Name, Salary, Level, Email, Birth Date, Country, City, Street Name, Donor ID) |

### 2nd NF

The Second Normal Form (2NF) builds upon the foundation of the First Normal Form (1NF) and addresses partial dependencies. For a relation to be in 2NF, it must first satisfy all 1NF requirements. Additionally, all non-key attributes must be fully functionally dependent on the entire primary key, ensuring that there are no partial dependencies. This means that every non-key attribute should depend solely on the whole primary key and not on a subset of it. The table below identifies any violations of 2NF in the schema and details the steps taken to resolve them, ensuring a higher level of normalization and minimizing redundancy:

|  |  |  |  |
| --- | --- | --- | --- |
| Relations | FDs | Violation description | Solution – Relations |
| Donor (Donor ID, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name) | Donor ID, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name | The Donor table, with attributes Donor ID, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, and Street Name, is in Second Normal Form (2NF). It is already in 1NF, and its primary key, Donor ID, uniquely identifies each record. All non-key attributes are fully functionally dependent on the primary key, with no partial dependencies, as there are no composite keys. | **Donor** (Donor ID, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name) |
| Event (Event ID, Event Name, Event Date, Venue Name, City, Street Name) | Event ID, Event Name, Event Date, Venue Name, City, Street Name | The Event table, with attributes Event ID, Event Name, Event Date, Venue Name, City, and Street Name, is in Second Normal Form (2NF). It satisfies 1NF, and its primary key, Event ID, uniquely identifies each record. All non-key attributes are fully functionally dependent on the primary key, with no partial dependencies, as there are no composite keys. | **Event** (Event ID, Event Name, Event Date, Venue Name, City, Street Name) |
| Donation (Donor ID, Amount, Date) | Donor ID, Amount, Date | The Donation table, with attributes Donor ID, Amount, and Date, is in Second Normal Form (2NF). It satisfies 1NF, with a composite primary key (Donor ID, Date). All non-key attributes (Amount) are fully functionally dependent on the composite primary key, and there are no partial dependencies | **Donation** (Donor ID, Amount, Date) |
| Staff (Staff ID, Staff First Name, Staff Last Name, Salary, Level, Email, Birth Date, Country, City, Street Name, Donor ID) | Staff ID, Staff First Name, Staff Last Name, Salary, Level, Email, Birth Date, Country, City, Street Name, Donor ID | The **Staff** table, with attributes Staff ID, Staff First Name, Staff Last Name, Salary, Level, Email, Birth Date, Country, City, Street Name, and Donor ID, is in Second Normal Form (2NF). It satisfies 1NF, and its primary key, Staff ID, uniquely identifies each record. All non-key attributes are fully functionally dependent on the primary key, with no partial dependencies, as there are no composite keys. | **Staff** (Staff ID, Staff First Name, Staff Last Name, Salary, Level, Email, Birth Date, Country, City, Street Name, Donor ID) |

### 3rd NF

3NF further extends the definition of 2NF by eliminating transitive dependencies. For a relation to be in 3NF, it needs to be in 2NF, and any dependency of a non-key attribute is to occur only with candidate keys, never with other non-prime attributes. That is to say, all the attributes of the table have to depend upon the primary key alone for ensuring data integrity and avoiding data redundancy. . The table below evaluates each relation for 3NF compliance, identifies any transitive dependencies, and provides solutions to restructure the schema accordingly:

|  |  |  |  |
| --- | --- | --- | --- |
| Relations | FDs | Violation description | Solution – Relations |
| The relations schema | Show the functional dependencies causing the violation | Describe why it is not in the 3rd NF (the violation) | Show the schema for each affected relation. |
| Donor (Donor ID, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name) | Donor ID, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name | The Donor table is in Third Normal Form (3NF). It satisfies 2NF because all non-key attributes are fully dependent on the primary key, Donor ID. Furthermore, there are no transitive dependencies, as each attribute depends directly on the primary key and not on other non-key attributes. Therefore, the Donor table adheres to 3NF. | **Donor** (Donor ID, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name) |
| Event (Event ID, Event Name, Event Date, Venue Name, City, Street Name) | Event ID, Event Name, Event Date, Venue Name, City, Street Name | The Event table is in Third Normal Form (3NF). The table meets the requirement for 2NF since all the non-key attributes are fully dependent on the primary key, Event ID. There are no transitive dependencies since each attribute directly depends on the primary key and not on any other non-key attributes. Hence, the Event table is in 3NF. | **Event** (Event ID, Event Name, Event Date, Venue Name, City, Street Name) |
| Donation (Donor ID, Amount, Date) | Donor ID, Amount, Date | The Donation table is in Third Normal Form. Since it has a composite primary key of Donor ID and Date that determines the non-key attribute of Amount, it fulfills the 2NF requirements. In addition, no transitive dependencies are there because all the attributes will directly depend on the primary key. Hence, the Donation table is in 3NF. | **Donation** (Donor ID, Amount, Date) |
| Staff (Staff ID, Staff First Name, Staff Last Name, Salary, Level, Email, Birth Date, Country, City, Street Name, Donor ID) | Staff ID, Staff First Name, Staff Last Name, Salary, Level, Email, Birth Date, Country, City, Street Name, Donor ID | The Staff table is in Third Normal Form. It follows 2NF because all the non-key attributes are fully dependent on the primary key, which is Staff ID. There are no transitive dependencies, as all non-key attributes, for instance, Country and City, depend directly on the primary key and not on any other non-key attributes. Thus, the Staff table is in 3NF. | **Staff** (Staff ID, Staff First Name, Staff Last Name, Salary, Level, Email, Birth Date, Country, City, Street Name, Donor ID) |