Detailed Design Document

1. Introduction-

Objective-

The Objective of this project is to make a Java Desktop application where users can come donate money for different charities.

Scope-

The Scope of this Project will be the users will be able to donate different amounts of funds, the information which will be stored in My SQL Database. Later (We could additionally add third party payment APIs so that users have a secure and easy time to donate money). We will have a Fundraiser Dashboard where different campaigns can be managed and an Admin Dashboard where Admin can control different users' account, update their information and can generate reports.

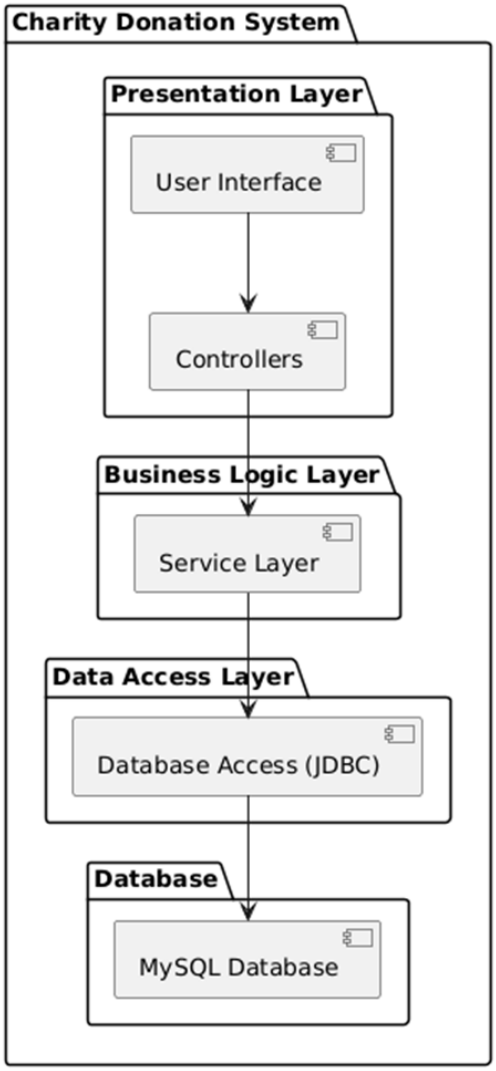
Assumptions-

It is assumed that user's information will be stored in My SQL only. Users will donate through the app only and not from their phone.

2. System Architecture Design

The Charity Donation System (CDS) is a **JavaFX-based desktop application** following a **3-tier architecture**:

1. **Presentation Layer** – JavaFX GUI.
2. **Business Logic Layer** – Core application logic.
3. **Data Access Layer (JDBC)** – Database operations with MySQL.



This diagram shows the architecture Design of this project.

3. Detailed Module Descriptions

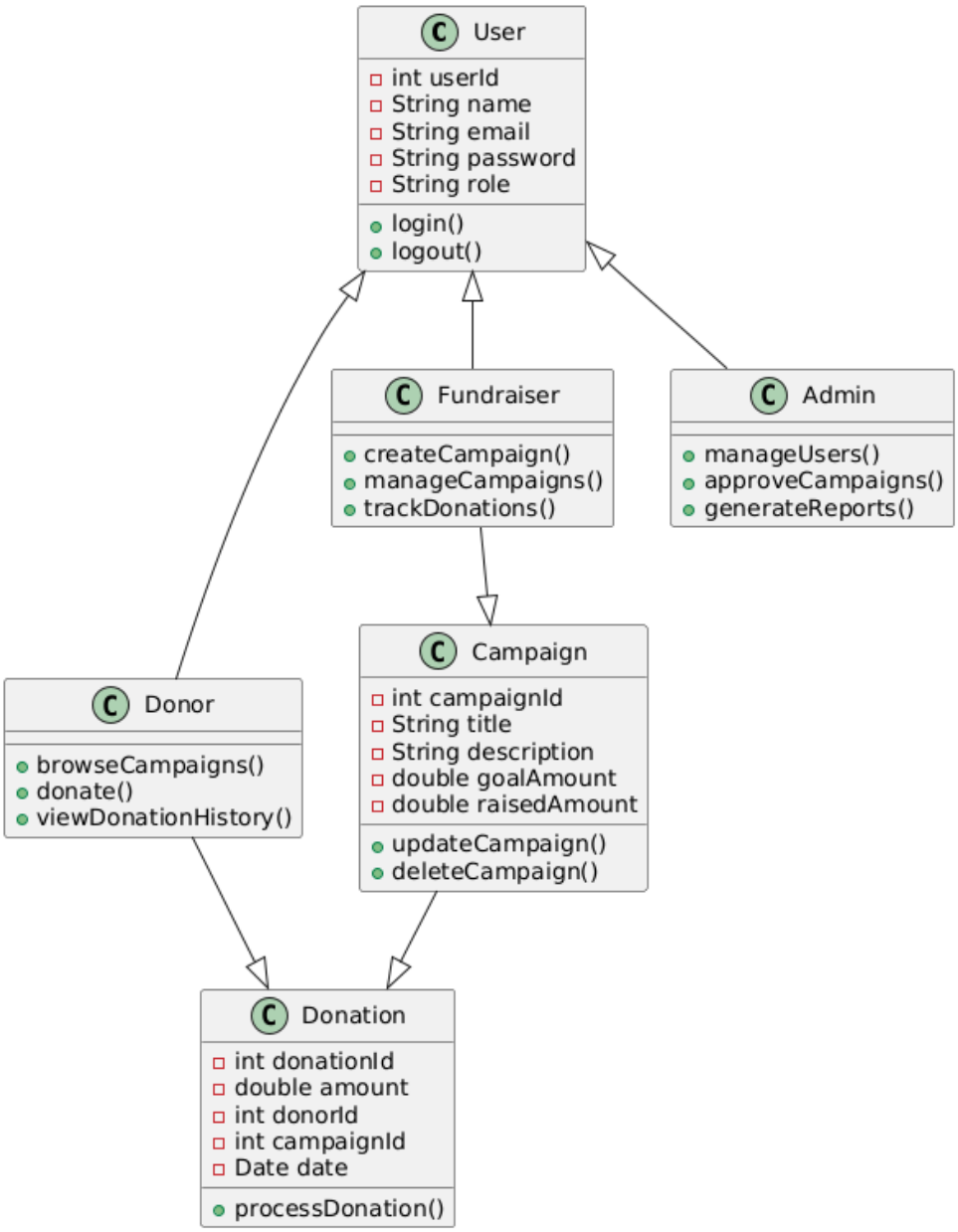
Presentation Layer- It is just UI how the will the app looks like, where the buttons should be, what font, color etc.

Business Logic Layer and Data Access Layer and Database-

This is the main part where users donate money to their charity selected, after that information is stored in the database, with the amount they provided, and payment information verification is done in the business Logic Layer.

4. Database Design

The E-R Diagram below shows how the database is connected to the other parts of the project.



Below is how actually in MY SQL Databases are made.

```
CREATE TABLE User (
  user_id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(100) NOT NULL,
```

```
 email VARCHAR(100) UNIQUE NOT NULL,
 password VARCHAR(255) NOT NULL,
 role ENUM('Donor', 'Fundraiser', 'Admin') NOT NULL
);
```

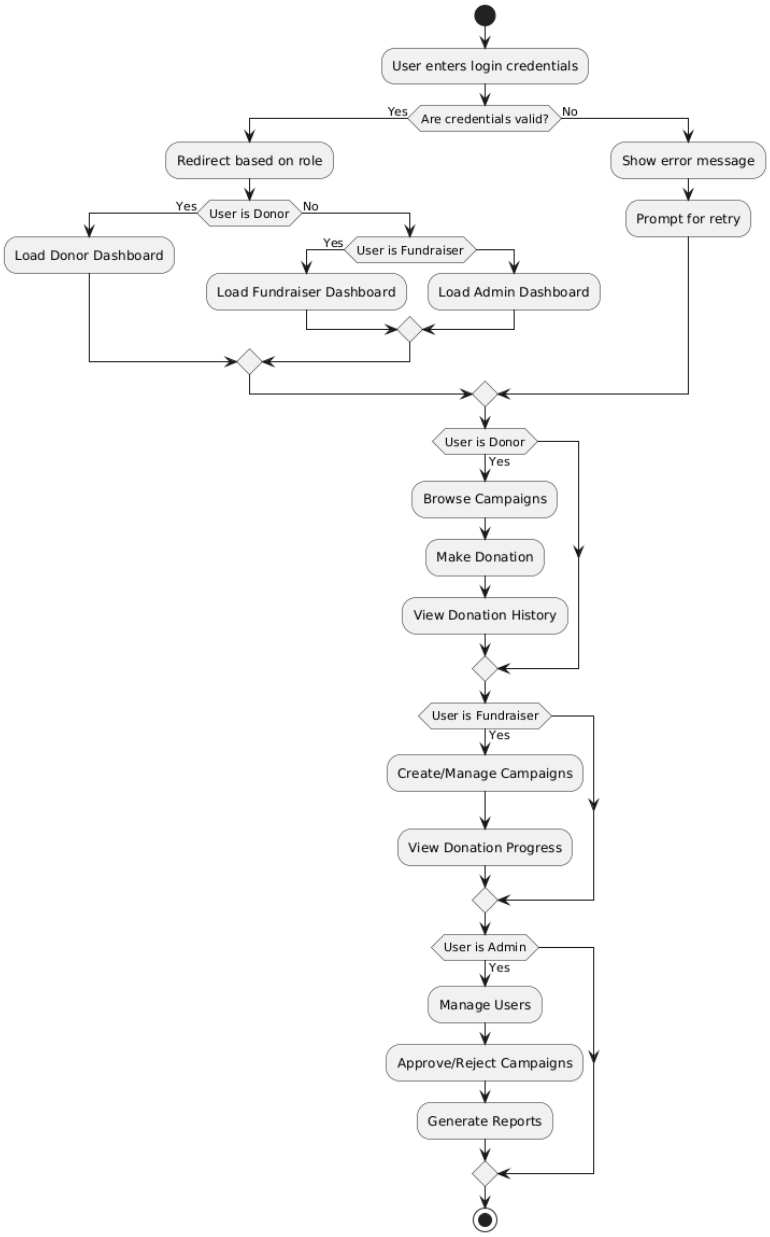
Campaign Table

```
CREATE TABLE Campaign (
 campaign_id INT PRIMARY KEY AUTO_INCREMENT,
 title VARCHAR(255) NOT NULL,
 description TEXT,
 goal_amount DECIMAL(10,2) NOT NULL,
 raised_amount DECIMAL(10,2) DEFAULT 0,
 fundraiser_id INT,
 FOREIGN KEY (fundraiser_id) REFERENCES User(user_id)
);
```

Donation Table

```
CREATE TABLE Donation (
 donation_id INT PRIMARY KEY AUTO_INCREMENT,
 amount DECIMAL(10,2) NOT NULL,
 donor_id INT,
 campaign_id INT,
 date DATE NOT NULL,
 FOREIGN KEY (donor_id) REFERENCES User(user_id),
 FOREIGN KEY (campaign_id) REFERENCES Campaign(campaign_id)
);
```

5. System Flow Design-



7. Interaction Between Components-

This part has already been shown in earlier parts please refer the E-R Diagram shown in 4 steps.

8. Non-Functional Requirements-

Non-Functional Requirements may Include in our project such as-
Security Measures- We can later with more budget add third party vendors to secure our payment system including teams such as IT and Cyber security specialists.

Also, we can do **Role-Based Access Control**: Restrict UI elements based on user roles.

Scalability-

Our project can be grown first by making it a full-grown website and improving our databases to account for more new data, improving our UI.

Reliability-

Our System will be strongly reliable as the user's information will be copied and stored in at least 2-3 different databases, and to recover something we can follow certain Digester recovering techniques and we can follow SQL Injection Technique.

Fast-

We could process donation processes faster by improving our system internally with new APIs or paid versions.