National Teachers College
629 J Nepomuceno, Quiapo, Manila, 1001 Metro Manila
Bachelor of Science in Information Technology

Final Project Documentation: Database Management System 1

DBMS for Sustainable Development

A Final Project
Presented to the Faculty of the
College of Information Technology

In partial fulfillment
of the Course Requirements for the degree
of Bachelor of Science in Information Technology

Submitted by:
Jericho B. Busa
Jeric Jay M. Entero
Keanu Sean G. Gabuya
Angelyn Santos

2.3 BSIT

National Teachers College

Instructor:
Ms. Justin Louise R. Neypes

Date:
December 12, 2025

I. Introduction

1.1. Project Overview & UN SDG Target

This project targets Sustainable Development Goal (SDG) 1: No Poverty, that is, reducing the economic vulnerability through providing equitable access to the social welfare programs to the people living in poverty. On this end, our team came up with the Relational Database Management System (RDBMS) that assists organizations to trace beneficiaries, process program enrollment and keep track of offering services that alleviate poverty like cash assistance, livelihood training, food support, healthcare subsidies and educational support.

The RDBMS allows effective data management by use of normalization, constraint implementation, stored program logic and automatic triggers. The system enables organizations to assess the efficiency of the programs and enhance the process of providing assistance to the population living in poverty by using structured data storage and complex SQL reporting.

1.2. Problem Statement

Poverty reduction organizations are usually faced by disseminated data, paper-based record keeping and reporting mismatch. All these problems result in multiple records of beneficiaries, unconfirmed eligibility, ineffective delivery of programs and inability to produce meaningful reports that can quantify the actual impact of poverty-alleviation programs.

In the absence of a centralized system, it is difficult to:

- Trace the identity of those people who are enrolled to which assistance program.
- Assure integrity of data among beneficiaries, programs and distributions.
- Keep track of the quantity and rate of assistances provided.
- Produce correct analytics and SDG-oriented reports that help in decision-making.

Our RDBMS aims to eliminate these issues by creating a structured and reliable platform that can store and manipulate records of beneficiaries, program data, and assistance transactions based on the principles of relational data storage, foreign key constraints, stored procedures, and automated triggers, which are all shown to be ACID compliant. This guarantees proper data, reduced redundancy and valid entries, and relevant reporting on SDG 1 initiatives.

II. Requirements & Analysis

2.1 Functional Requirements and Non-Functional Requirements

Functional Requirements (FR)

ID	Requirement
FR1	**DDL/Schema:** System needs to be able to load and be able to store input data in the complete designed relational schema (Beneficiary, Program, AssistanceRecord).
FR2	**Integrity:** System should implement valid foreign key relationships, NOT NULL and CHECK constraints to all the core tables to ensure data integrity.
FR3	**ACID SP:** System should invoke a transactional Stored Procedure (SP_DisburseAssistance) which will explicitly enforce ACID properties of capturing assistance events.
FR4	**Triggers:** System should have a Trigger (T_UpdateTotalAid) that will automatically update the Beneficiary record, depending on the AssistanceRecord insertion.
FR5	**Reports/DQL:** System should produce two or more (2) different reports associated with SDG 1 impact through an intricate SQL query through VIEWS.
FR6	**DCL:** System has to use user roles and permissions via DCL (GRANT/REVOKE statements).

Non-functional Requirements (NFR)

NFR1	Reliability: FR3 core transactional logic is required to ensure Durability and Atomicity.
NFR2	Maintainability: The whole database model should be documented, modular and normalized to 3NF at least.
NFR3	Performance: The complex report queries (FR5) should have a high performance with results of less than 1 second using sample data.
NFR4	Robustness: The system has to be able to deal with invalid input amicably through the use of database level constraints and informative error messages (e.g. in the Stored Procedure).

2.2 Data Requirements

The RDBMS will manage data across three core entities. Key data fields include:

- **Beneficiary Data**: BeneficiaryID, ID_Number (unique identifier), FullName, DateOfBirth, HouseholdSize, AnnualIncome, City.
    - **Fields which are managed by the system:** TotalAidReceived (automatically updated by a Trigger to facilitate reporting) and RegistrationDate (system date which was used by auditing).
- **Program Data**: ProgramID, ProgramName (unique identifier), ProgramCategory (e.g., 'Cash Aid', 'Food'), Description.
- **Assistance Record Data (Transaction)**: BeneficiaryID (FK), ProgramID (FK), AmountGiven, DistributionDate.

Expected Data Volume (Test Assumptions)
The initial test environment will use generated data to validate integrity and performance.

Source: Data will be generated using **Mockaroo**.

Volume: A minimum of **50+ records** total, aiming for:

- ~ 100 Beneficiary Records
- ~ 10 Program Records (Predefined)
- ~ 500 Assistance Transaction Records (Test Data)

2.3 Schema Normalization Analysis

Core Tables

1. Beneficiary
2. Program
3. AssistanceRecord (Junction/M:N Table)

Normalization Justification

The database schema follows the third normal form (3NF) and the Boyce-Codd normal form (BCNF) in order to remove data duplication and anomaly.

- **1NF (First Normal Form):** Met by the fact that all the attributes are atomic (single-valued) and have no repeating groups.
- **2NF (Second Normal Form):** This is attained because all non-key attributes are entirely dependent on the whole Primary Key.
- **3NF (Third Normal Form):** This is attained since there are no transitive dependencies. Any other non-key attributes rely on the Primary Key.

Table	Primary Key (PK)	3NF/BCNF Status
Beneficiary	BeneficiaryID	3NF/BCNF. Each attribute is based on BeneficiaryID only.
Program	ProgramID	3NF/BCNF. Attributes are all based on the ProgramID.
AssistanceRecord	AssistanceID	3NF/BCNF. The Many-to-Many relationship has been properly applied in the table and all attributes rely on the transaction ID.

III. Design Specification

3.1 Core DBMS Concepts Used (The Five):

1. **Stored Procedure: "SP_DisburseAssistance"**

This Stored Procedure is used to access the disbursement process as a one-time atomic transaction. It maintains data consistency by applying a business rule: only those beneficiaries who have an AnnualIncome not greater than P25,000 can be offered assistance. It ensures safety in transactions by eliminating any error and ROLLBACK in case of failure in integrity check or system breakdown.

SQL CODE

```
-- CREATE PROCEDURE (SP_DisburseAssistance)
DELIMITER $$
CREATE PROCEDURE SP_DisburseAssistance (
    IN p_BeneficiaryID INT,
    IN p_ProgramID INT,
    IN p_DistributionDate DATE,
    IN p_AmountGiven DECIMAL(10, 2)
)
BEGIN
    DECLARE v_Income DECIMAL(10, 2);
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        -- Error handling: If anything fails, rollback the entire transaction
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Transaction failed due to system
error.';
    END;
```

```
    -- 1. BEGIN TRANSACTION
    START TRANSACTION;

    -- 2. Consistency Check (Example: Only disburse to those with annual income <= 25000)
    SELECT AnnualIncome INTO v_Income
    FROM Beneficiary
    WHERE BeneficiaryID = p_BeneficiaryID;

    IF v_Income > 25000.00 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Beneficiary is ineligible: Annual
income exceeds P25,000 threshold.';
        ROLLBACK;
    ELSE
        -- 3. Atomicity: Insert the new record
        INSERT INTO AssistanceRecord (BeneficiaryID, ProgramID, DistributionDate,
AmountGiven)
        VALUES (p_BeneficiaryID, p_ProgramID, p_DistributionDate, p_AmountGiven);

        -- The Trigger (T_UpdateTotalAid) executes automatically here.

        -- 4. COMMIT the transaction
        COMMIT;
    END IF;
END$$
DELIMITER ;
```

2. **Trigger: “T_UpdateTotalAid”**

The Trigger provides integrity of data and referential integrity. It is an automated mechanism that automatically updates the TotalAidReceived column in the Beneficiary table when a new record has been successfully inserted in the AssistanceRecord table. This avoids mistakes by the user and also makes sure that the running aid amount of the beneficiary is never mistaken.

SQL CODE

```
-- CREATE TRIGGER (T_UpdateTotalAid)
DELIMITER $$
CREATE TRIGGER T_UpdateTotalAid
AFTER INSERT ON AssistanceRecord
FOR EACH ROW
BEGIN
    -- Update the TotalAidReceived in the Beneficiary table
    UPDATE Beneficiary
    SET TotalAidReceived = TotalAidReceived + NEW.AmountGiven
```

```
    WHERE BeneficiaryID = NEW.BeneficiaryID;
END$$
DELIMITER ;
```

3. **Complex View**

This Complex V_ProgramSummary or V_TopBeneficiaries makes it easy to access data to do management reporting (FR5). It integrates the information of three tables with the help of aggregation (SUM and COUNT()) functions and GROUP BY statement to show real-time performance statistics without having the user enter a complex query.

SQL CODE

```
-- CREATE VIEW 1 (V_ProgramSummary)
-- Report 1: Total aid given per program, including the program's category.
CREATE VIEW V_ProgramSummary AS
SELECT
    p.ProgramName,
    p.ProgramCategory,
    COUNT(ar.AssistanceID) AS TotalDisbursements,
    SUM(ar.AmountGiven) AS TotalAidDisbursed
FROM
    Program p
JOIN
    AssistanceRecord ar ON p.ProgramID = ar.ProgramID
GROUP BY
    p.ProgramID, p.ProgramName, p.ProgramCategory
ORDER BY
    TotalAidDisbursed DESC;


-- CREATE VIEW 2 (V_TopBeneficiaries)
-- Report 2: List of beneficiaries who received the highest aid, including their demographic and
income details.
CREATE VIEW V_TopBeneficiaries AS
SELECT
    b.FullName,
    b.City,
    b.HouseholdSize,
    b.AnnualIncome,
    b.TotalAidReceived
FROM
    Beneficiary b
ORDER BY
    b.TotalAidReceived DESC
```

```
LIMIT 10;
```

4. **Complex Relationships: Many-to-Many (M:N)**

The Beneficiary-Program is Many-to-Many (M:N) and thus a single beneficiary may be offered help by a large number of programs and a single program may assist a large number of beneficiaries. This relationship is applied through the AssistanceRecord table which is a Junction Table that has Foreign Keys (BeneficiaryID and ProgramID).

SQL CODE

```
CREATE TABLE AssistanceRecord (
   AssistanceID INT PRIMARY KEY AUTO_INCREMENT,
   BeneficiaryID INT NOT NULL,
   ProgramID INT NOT NULL,
   DistributionDate DATE NOT NULL,
   AmountGiven DECIMAL(10, 2) NOT NULL,

   -- Mandatory Constraint: Ensure assistance is positive (FR2)
   CHECK (AmountGiven > 0.00),

   -- Foreign Keys enforce referential integrity (FR2 & Midterm Concept)
   FOREIGN KEY (BeneficiaryID) REFERENCES Beneficiary(BeneficiaryID) ON DELETE
CASCADE,
   FOREIGN KEY (ProgramID) REFERENCES Program(ProgramID) ON DELETE RESTRICT,

   -- Index for faster joins in complex reports (FR5)
   INDEX (BeneficiaryID, ProgramID)
);
```

5. **Integrity Constraints (CHECK/UNIQUE/FK)**

The system employs all the key integrity measures (PK, FK, UNIQUE, and CHECK) to implement data quality (FR2). There is a UNIQUE constraint on ID_Number and ProgramName to avoid duplication of the registration. The referential integrity is enforced on the M:N relationship by foreign (FKs). In addition, domain integrity is also guaranteed by the explicit CHECK constraints such as the absence of illogical data, including negative income (AnnualIncome >= 0), zero household size (HouseholdSize >= 1), and negative aid given (AmountGiven > 0.00).

SQL CODE

```
-- Program Table
ProgramName VARCHAR(100) NOT NULL UNIQUE, -- UNIQUE Constraint
CHECK (LENGTH(ProgramName) > 5) -- CHECK Constraint
```

```
-- Beneficiary Table
ID_Number VARCHAR(20) NOT NULL UNIQUE, -- UNIQUE Constraint
CHECK (AnnualIncome >= 0), -- CHECK Constraint
CHECK (HouseholdSize >= 1)  -- CHECK Constraint

-- AssistanceRecord Table
CHECK (AmountGiven > 0.00), -- CHECK Constraint
FOREIGN KEY (BeneficiaryID) REFERENCES Beneficiary(BeneficiaryID) ON DELETE
CASCADE, -- FOREIGN KEY Constraint
FOREIGN KEY (ProgramID) REFERENCES Program(ProgramID) ON DELETE RESTRICT --
FOREIGN KEY Constraint
```

6. **DCL: User Roles and Permissions (FR6)**

The system also provides a security layer that is based on Data Control Language (DCL) statements to establish user roles (Admin, Data_Entry, Reporter) and implement the least privilege principle. The mechanism will make sure that sensitive information cannot be changed by an unauthorized user, therefore, safeguarding the integrity of the system against human mistakes and ill intent.

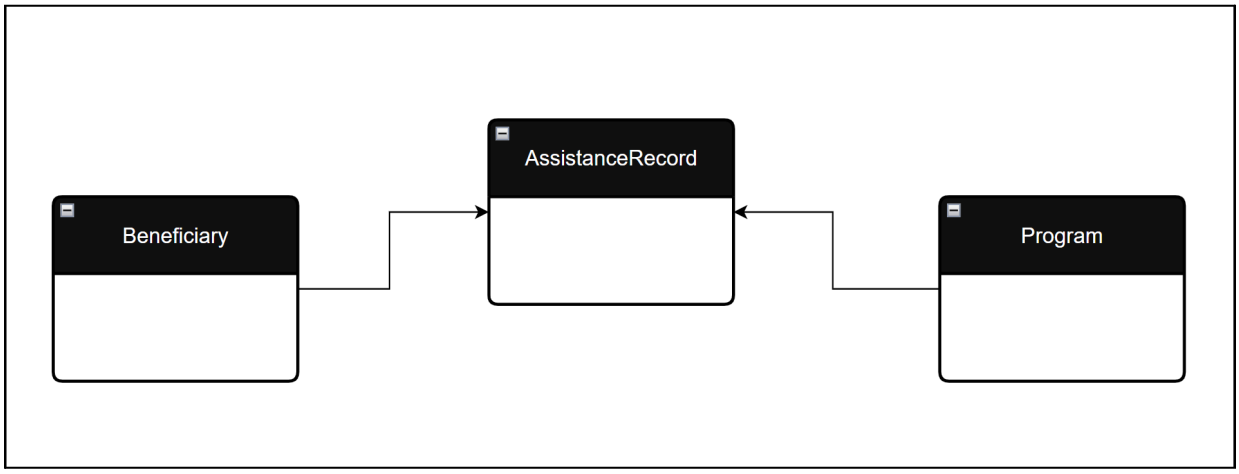
SQL CODE

```
-- 1. Create Users
CREATE USER 'Admin'@'localhost' IDENTIFIED BY 'secure_admin123';
CREATE USER 'Data_Entry'@'localhost' IDENTIFIED BY 'secure_data_entry123';
CREATE USER 'Reporter'@'localhost' IDENTIFIED BY 'secure_reporter123';

-- 2. Grant Permissions

-- Admin: Full control
GRANT ALL PRIVILEGES ON Poverty_Alleviation_System.* TO 'Admin'@'localhost' WITH
GRANT OPTION;

-- Data Entry: Can execute the SP and SELECT/INSERT into relevant tables
GRANT EXECUTE ON PROCEDURE SP_DisburseAssistance TO 'Data_Entry'@'localhost';
GRANT SELECT, INSERT ON AssistanceRecord TO 'Data_Entry'@'localhost';
GRANT SELECT ON Beneficiary TO 'Data_Entry'@'localhost';

-- Reporter: Read-Only access for reports
GRANT SELECT ON Beneficiary TO 'Reporter'@'localhost';
GRANT SELECT ON AssistanceRecord TO 'Reporter'@'localhost';
GRANT SELECT ON Program TO 'Reporter'@'localhost';
GRANT SELECT ON V_ProgramSummary TO 'Reporter'@'localhost';
GRANT SELECT ON V_TopBeneficiaries TO 'Reporter'@'localhost';
```

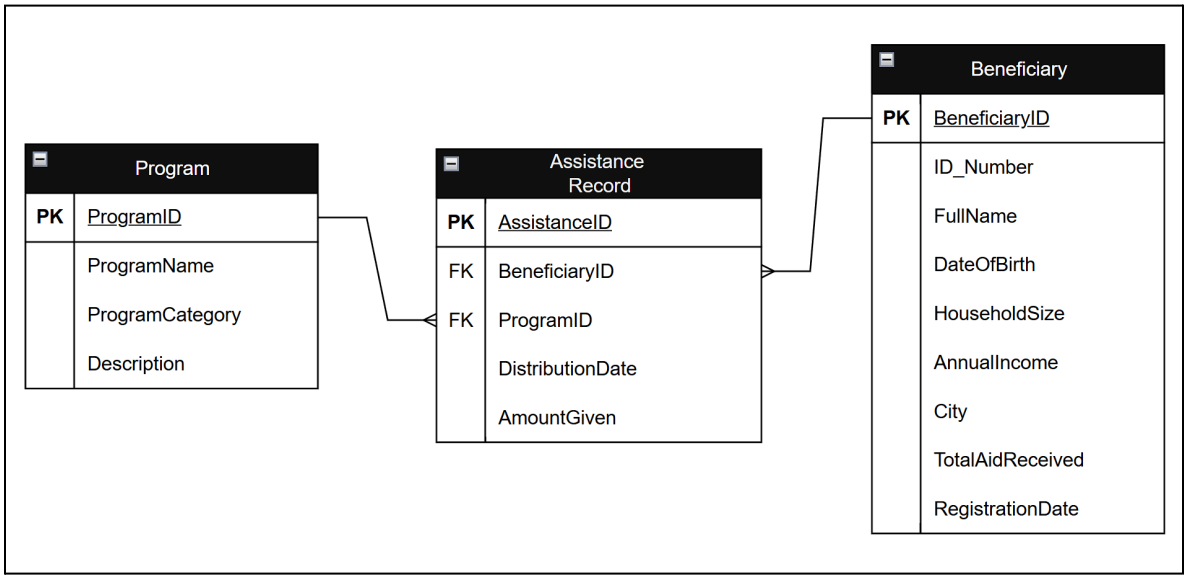
```
-- 3. Apply Changes (Crucial step for DCL commands to take effect immediately)
FLUSH PRIVILEGES;
```

3.2 ER Diagram

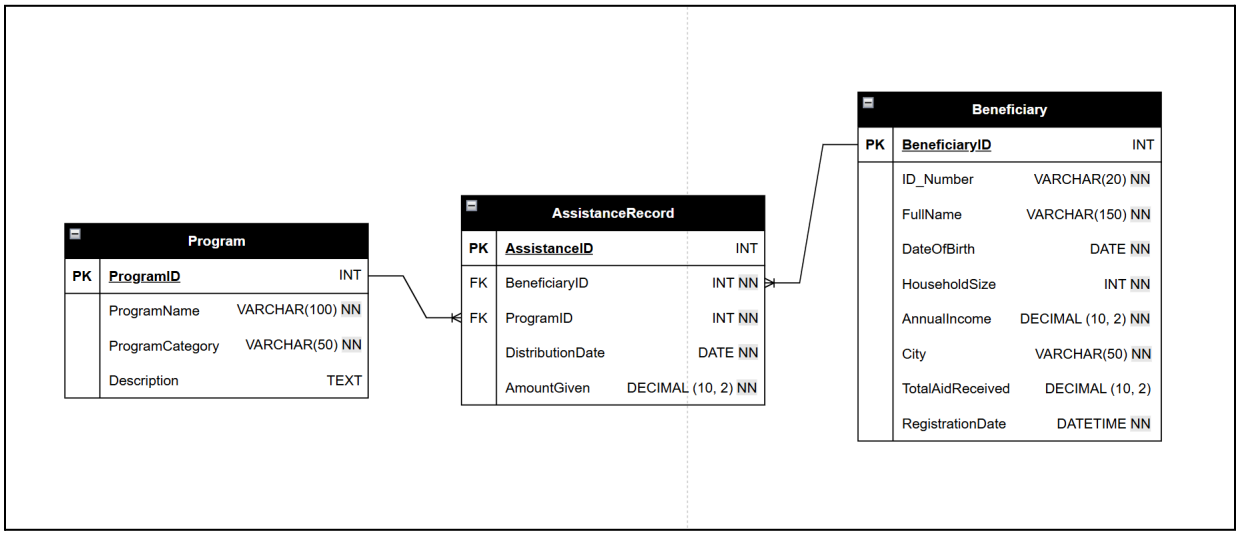
Conceptual



Logical

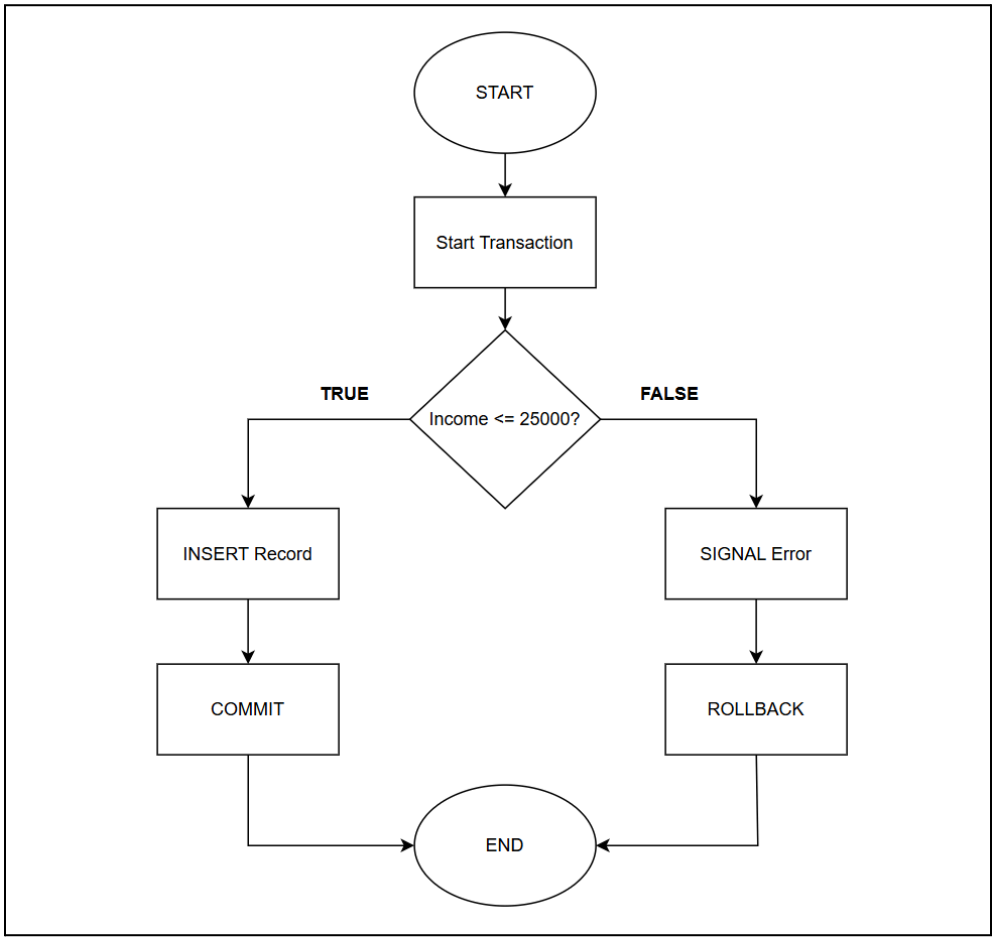


Physical



3.3 Transaction Flowchart

Stored Procedure Flowchart



V. Conclusion

The Poverty Alleviation Database Management System has been implemented and confirmed. The completed system satisfies all operational demands, having a secure, scalable, and reliable platform to track and administrate social aid disbursements.

Key Features

- **Data Integrity and Security (FR2, FR3)**: Stored Procedure (SP_DisburseAssistance) will ensure the ACID properties of each transaction which verifies the eligibility check (Consistency) and avoids partial data recording (Atomicity). The Trigger (T_UpdateTotalAid) is used to automate key total work, which maintains data integrity.
- **System Structure**: A normalized data model was created, and properly a Many-to-Many relationship between the Beneficiaries and Programs was implemented through the junction table.
- **Reporting (FR5)**: Complex Views: Complex Views enable the management to see an immediate, aggregated report on program effectiveness and performance of beneficiaries without dealing with raw data to present an actionable report.

The system is strictly compliant and has been tested successfully to prove both committed transactions and intentional ROLLBACK failures.

Member's Contribution

Member Focus	**Deliverables & Responsibilities** **Design and implementation of the:**	**Specific Code/Documentation Contributed**
Jericho B. Busa (SP & ACID)	**Stored Procedure Implementation**: The core transactional procedure to enforce ACID properties and eligibility rules.	• SP_DisburseAssistance SQL code (**1.3_StoredLogic.sql**) • Transaction Flowchart (**3.3**) • Justification for the Stored Procedure (**3.1**)
Keanu Sean G. Gabuya (Trigger & DML)	**Trigger Implementation & Data Setup**: Data integrity trigger. Responsible for preparing the DML script and ensuring final data accuracy.	• T_UpdateTotalAid SQL code (**1.3_StoredLogic.sql**) • DML Test Data Generation (**1.2_DML_TestData.sq**l) • Justification for the Trigger (**3.1**)

Angelyn Santos (Views & Reporting)	**Complex Views Implementation**: Complex queries for management reporting (FR5). Responsible for final presentation preparation.	● V_ProgramSummary and V_TopBeneficiaries SQL code (**3_StoredLogic.sql**) ● Justification for the Complex VIEW (**3.1**) ● Testing and Results documentation (**4.1** and **4.2**)
Jeric Jay M. Entero (DDL & Schema)	**Schema Design & Structure**: Database structure, table definitions, and relationships. Responsible for final ERD generation.	● Database Schema (DDL) (**1.1_DDL_Schema.sql**) ● Definition of the M:N Relationship (**3.1**) ● Final Physical ERD Generation (**3.2**) ● Justification for Integrity Constraints (**3.1**) ● Data Control Language (FR6) **(1.4_DCL_Permission.sql)**