Lend a Hand

May 2025

Dean Feldman-2803899

Liora Rosenberg-2693084

Nadav Sundy- 2698600

Shayna Unterslak-2799578

Contents

[Description of the procedures and processes involved in the Lend a Hand Project 3](#_Toc196742171)

[User Login in 3](#_Toc196742172)

[User Sign up 3](#_Toc196742173)

[Forgot Password 3](#_Toc196742174)

[Donor Wall of Fame 3](#_Toc196742175)

[Profile page 4](#_Toc196742176)

[Donate Page 4](#_Toc196742177)

[Receiver Page 4](#_Toc196742178)

[Business Rules 4](#_Toc196742179)

[Development of initial ERD 5](#_Toc196742180)

[Recognition of issues in the initial ERD 7](#_Toc196742181)

[Improved ERD 7](#_Toc196742182)

[Implementation of Tables 9](#_Toc196742183)

[USER Table 9](#_Toc196742184)

[ITEM Table 9](#_Toc196742185)

[REQUEST Table 9](#_Toc196742186)

[DONATION Table 10](#_Toc196742187)

## Description of the procedures and processes involved in the Lend a Hand Project

User Login in

* The user will be asked to log in
* The user will log in using email address and password using the Log in Button
* The user input will be validated against the USER table in the database
* If the input is valid, the user will be able to view the Donor Wall of Fame
* If the user forgot their password, there will be a Forgot Password button
* The user will be able to select a sign up option if they do not have an account which will open a sign up page

User Sign up

* The user will enter their:
  + First name
  + Surname
  + Date of birth
  + Email address
  + Password
  + Confirm Password
* The input will be validated:
  + An account with the entered email address doesn’t exist
  + If an account with this email address exists, the user will be informed and must return to Log in screen if that is the email they want to use
  + The email address will be verified:
    - It must contain an @ symbol
    - It cannot have spaces
  + The email address will be verified with a One-Time-Pin
  + The password must meet the following requirements:
    - At least 1 upper case letter
    - At least 1 lower case letter
    - At least 1 symbol
    - At least 6 characters long
  + The user must be over 18

Forgot Password

* The user will enter their email address
* The user will receive a One-Time-Pin to validate that they are the one changing their password
* The user will be able to change their password

Donor Wall of Fame

* The user will be able to view all donors ordered by the number of items donated

Profile page

* The user will view all their personal information
* The user will be able to add a short biography about themselves as a motivation for items they are in need of.
* The user will be able to edit their personal information (excluding email addresses) which will be verified following the same verification as sign up

Donate Page:

* The users will select the item they want to donate from a drop down menu
* The user will select a quantity that they are willing to donate
* The user will be shown a list of receivers (users that need this specific item) and the quantity they require
* The user will be able to view the receiver’s biography (if the receiver has one)
* The user will decide who they want to donate to
* The user will get the contact details of the person they will donate to so that the donation can take place.
* The quantity that the user can donate, will decrease
* The quantity that the receiver needs will decrease in the REQUEST table in the database
* The user will be added to or updated in the donor wall
* The user will be able to select another item to donate

Request Page:

* The users will select the item they need from a drop down menu
* The user will select a quantity that they are looking for
* These details will be added to the REQUEST table in the database
* The user will be able to select other items should they need them

## Business Rules

* Each user must have a unique email address
* A user must be at least 18 years old to register
* A user password must be at least 6 characters, with one uppercase letter, lowercase letter and symbol
* Passwords must be stored securely (hashed)
* A user that has donated an item will appear on the Donor Wall of Fame. The Donor Wall of Fame will appear in order of amount donated.
* A donor must appear on the Donor Wall of Fame after their first donation.
* All users must be able to view the Donor Wall of Fame
* A user will be able to add a biography about themselves which the donor can view before deciding who to donate to. This biography is not required.
* Users can only edit their profile information except for their email address.
* Users will only be able to receive or donate from a predefined list.
* The predefined list of items is:
  + Tinned Tuna
  + Rice
  + Maize Meal
  + Peanut butter
  + Bread
  + Eggs
  + Toilet Paper
  + Soap
  + Sanitary Pads
  + Baby Formula
  + Nappies
  + Blanket
  + Pens
  + Pencil
  + Eraser
  + Ruler
* A user can request multiple items
* An item can be requested by multiple users
* A user can donate multiple items
* An item can be donated by multiple users
* A donation must reduce the donor’s available quantity and the receiver's needed quantity.
* A donation must link a donor user, a receiver user, and an item.
* If the needed quantity reaches 0, the request is considered fulfilled and is removed from the database

## Development of initial ERD

The organisation depicted in the ERD has 4 entities

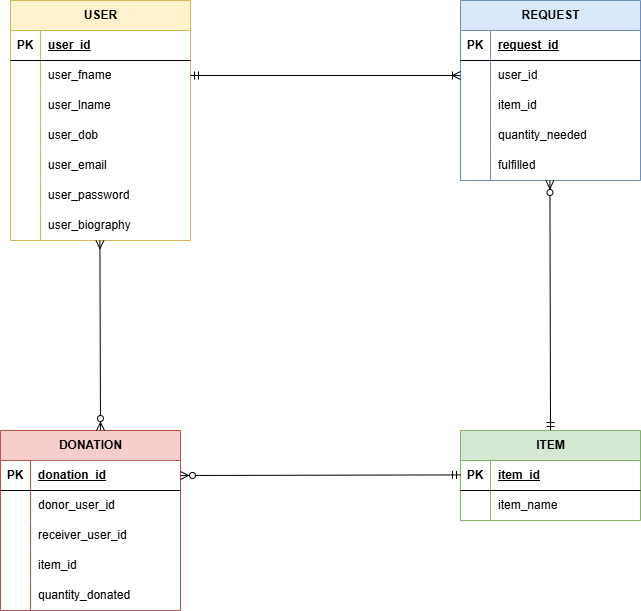
* USER – information about each user
* ITEM- information about each item that can be donated
* REQUEST- information about the request for items a user makes
* DONATION- information about a donation that takes place

Each entity has the following attruibutes:

* USER
  + user\_id (primary key)
  + user\_fname
  + user\_lname
  + user\_dob
  + user\_email (Unique)
  + user\_password (hashed)
  + user\_biography (nullable)
* ITEM
  + item\_id (primary key)
  + item\_name
* REQUEST
  + request\_id (primary key)
  + user\_id (foreign key to USER)
  + item\_id (foreign key to ITEM)
  + quantity\_needed
  + fulfilled (Boolean)
* DONATION
  + donation\_id (primary key)
  + donor\_user\_id (foreign key to USER)
  + receiver\_user\_id (foreign key to USER)
  + item\_id (foreign key to ITEM)
  + quantity\_donated

Relationships:

* 1 user can make many requests
* Each request can be made by 1 user
* Each request can be for a single item
* Each item can be requested many times
* Each item can be donated many times
* Each donation consists of a single item
* Each donation includes many users, one that donates and one that receives
* Each user can be involved in many donations



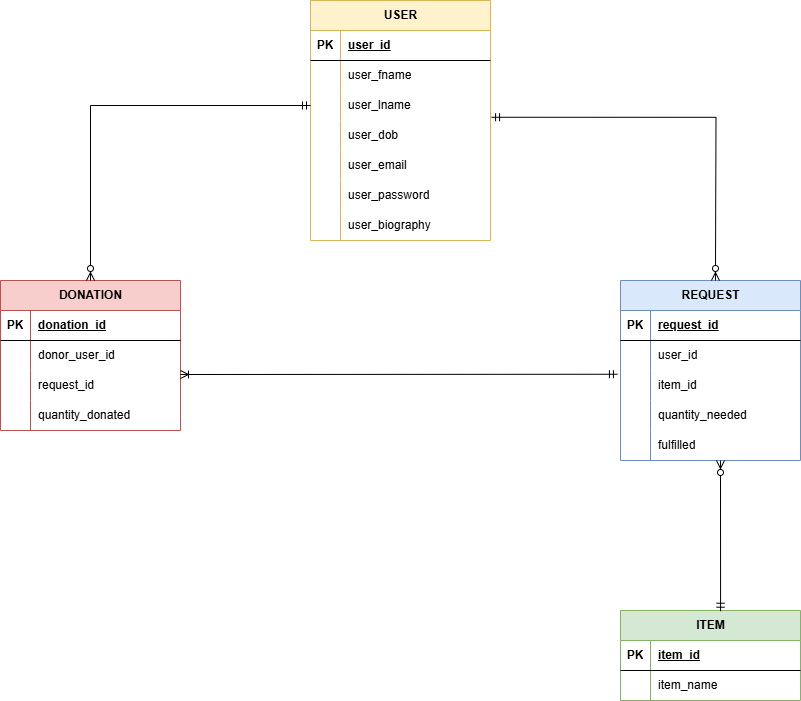
## Recognition of issues in the initial ERD

A Many-to-Many (M:N) Relationship exists between USER and DONATION.  
This is resolved by using the DONATION table as a bridge entity that links a donor user and a receiver user for each donation transaction.  
Each donation also links to a specific ITEM, ensuring that the association between users and items remains normalized and structured.  
The use of the DONATION table avoids any direct M:N relationship between users themselves and ensures that each donation event is properly recorded with details like quantity and date.

This ERD does not make use of any multivalued attributes.  
Each request and donation is stored as a separate record, allowing users to request or donate multiple items without violating normalization principles.

In the USER table, the user\_biography field may have NULL entries as it is optional.  
This is necessary and unavoidable for this project to allow users flexibility in providing personal motivation stories for donors.

## Improved ERD



The organisation depicted in the ERD has 4 entities

* USER – information about each user
* ITEM- information about each item that can be donated
* REQUEST- information about the request for items a user makes
* DONATION- information about a donation that takes place

Each entity has the following attruibutes:

* USER
  + user\_id (primary key)
  + user\_fname
  + user\_lname
  + user\_dob
  + user\_email (Unique)
  + user\_password (hashed)
  + user\_biography (nullable)
* ITEM
  + item\_id (primary key)
  + item\_name
* REQUEST
  + request\_id (primary key)
  + user\_id (foreign key to USER)
  + item\_id (foreign key to ITEM)
  + quantity\_needed
  + fulfilled (Boolean)
* DONATION
  + donation\_id (primary key)
  + donor\_user\_id (foreign key to USER)
  + request\_id (foreign key to REQUEST)
  + quantity\_donated

Relationships:

* 1 user can make many requests
* Each request can be made by 1 user
* Each request can be for a single item
* Each item can be requested many times
* Each item can be donated many times
* Each donation consists of a single item
* Each donation is directed at one request
* Each request is fulfilled by one or many donation

It is important to note, that if a user only donates part of their items to a user, the other items being donated will be considered a separate donation completely.

Following the development of the updated ERD, the database will be implemented using a relational database management system (RDBMS) using MySQL. Tables for USER, ITEM, REQUEST, and DONATION will be created according to the improved design, with appropriate primary keys, foreign keys, and constraints to enforce referential integrity.

Passwords will be securely stored using hashing techniques to ensure user security.  
The user\_biography field will be set as nullable to allow flexibility during user profile creation.

Default values will be set where necessary (such as the fulfilled status in REQUEST defaulting to 'false').

Stored procedures or triggers may also be developed to automate certain behaviors, such as updating the quantity\_needed field when a donation is made or removing fulfilled requests.

Overall, the database implementation will closely follow the final ERD to ensure a normalized, efficient, and secure system ready to support the Lend a Hand mobile application.

## Implementation of Tables

### USER Table

CREATE TABLE USER (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

date\_of\_birth DATE NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

biography TEXT

);

### ITEM Table

CREATE TABLE ITEM (

item\_id INT PRIMARY KEY AUTO\_INCREMENT,

item\_name VARCHAR(100) NOT NULL

);

### REQUEST Table

CREATE TABLE REQUEST (

request\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

item\_id INT NOT NULL,

quantity\_needed INT NOT NULL,

fulfilled BOOLEAN DEFAULT FALSE,

FOREIGN KEY (user\_id) REFERENCES USER(user\_id),

FOREIGN KEY (item\_id) REFERENCES ITEM(item\_id)

);

### DONATION Table

CREATE TABLE DONATION (

donation\_id INT PRIMARY KEY AUTO\_INCREMENT,

donor\_user\_id INT NOT NULL,

receiver\_user\_id INT NOT NULL,

item\_id INT NOT NULL,

quantity\_donated INT NOT NULL,

donation\_date DATE NOT NULL,

FOREIGN KEY (donor\_user\_id) REFERENCES USER(user\_id),

FOREIGN KEY (receiver\_user\_id) REFERENCES USER(user\_id),

FOREIGN KEY (item\_id) REFERENCES ITEM(item\_id)

);

**??? implementation of tables/functions/procedurs/views/triggers needed for the projet???**

**Stored Procedure: MakeDonation**

This procedure handles the donation transaction: inserting a new donation record, updating the quantity needed by the receiver, and marking requests as fulfilled if needed.

DELIMITER //

CREATE PROCEDURE MakeDonation(

IN p\_donor\_id INT,

IN p\_receiver\_id INT,

IN p\_item\_id INT,

IN p\_quantity INT

)

BEGIN

-- Insert the donation

INSERT INTO DONATION (donor\_user\_id, receiver\_user\_id, item\_id, quantity\_donated, donation\_date)

VALUES (p\_donor\_id, p\_receiver\_id, p\_item\_id, p\_quantity, CURDATE());

-- Update the quantity needed in the REQUEST

UPDATE REQUEST

SET quantity\_needed = quantity\_needed - p\_quantity,

fulfilled = CASE WHEN quantity\_needed - p\_quantity <= 0 THEN TRUE ELSE FALSE END

WHERE user\_id = p\_receiver\_id AND item\_id = p\_item\_id;

END //

DELIMITER ;

**View: DonorWall**

This view displays all users who have donated, sorted by the total number of items donated in descending order. This supports the Donor Wall of Fame feature in the app.

sql

CopyEdit

CREATE VIEW DonorWall AS

SELECT

u.first\_name,

u.last\_name,

SUM(d.quantity\_donated) AS total\_donated

FROM

USER u

JOIN

DONATION d ON u.user\_id = d.donor\_user\_id

GROUP BY

u.user\_id

ORDER BY

total\_donated DESC;

**Trigger: CheckRequestFulfillment**

This trigger automatically updates a request’s status to fulfilled once the quantity needed reaches zero after a donation.

sql

CopyEdit

DELIMITER //

CREATE TRIGGER CheckRequestFulfillment

AFTER UPDATE ON REQUEST

FOR EACH ROW

BEGIN

IF NEW.quantity\_needed <= 0 THEN

UPDATE REQUEST

SET fulfilled = TRUE

WHERE request\_id = NEW.request\_id;

END IF;

END //

DELIMITER ;