Project

Ali USTUNYER

Title:

Anonymized Utility Bill Support Platform

Executive Summary:

The Anonymized Utility Bill Support Platform is a nationwide web-based solution designed to facilitate philanthropic aid for individuals in need throughout Turkey by enabling them to anonymously submit their utility bills (water, electricity, natural gas, etc.) for financial assistance. Donors can browse these bills and pay them based on unique ID numbers, thereby preserving the beneficiaries' dignity and eliminating social stigma. This innovative system eliminates the need for direct monetary transactions between parties and promotes a culture of social responsibility and solidarity within the country. This system also aims to alleviate the financial burden on those in need, while preserving their dignity by maintaining their anonymity.

The project involves creating a user-friendly, secure, and scalable web platform, and initial features will focus on user registration, utility bill submission and management, payment processing, and basic reporting.

Initial Features:

List of Features/Endpoints for each feature:

- Entities: 1-Beneficiary, 2- Donor 3-UtilityBills, 4-Payments

1-Beneficiary: Stores beneficiary information.

2- Donor: Stores donor information

3-Utility_bills: Stores utility bill information.

4-Payments: Stores payment information (join table for many-to-many relationship.

UtilityBills *table- Users can perform the following operations:*

◦ View submitted utility bills (GET on UtilityBills by utility type and amount due)

Donors table – Users can perform the following operations:

- Create users (donors) with a POST operation.

Beneficiaries table – Users can perform the following operations:

- Create users (beneficiaries) with a POST operation.
- View beneficiaries with a GET operation (by last_name).
- Update beneficiary information with a PUT operation (using beneficiary_id)
- Delete beneficiaries with a DELETE operation (using beneficiary_id)

Payments table (Users can perform the following operations:

◦ Create payments with a POST operation.*

*When the payment is created the relevant row in the utilityBills is also UPDATED as “isPaid” accordingly.

*All the operations have spesific test classes and they work in accordance with the spesifications of each operation.

Stretch Goals (to be completed if time allows, or after graduation):

Beneficiaries *can perform the following operations:*

◦ Submit a utility bill (POST on UtilityBills)

◦ Update their submitted utility bills (PUT on UtilityBills by bill_id)

◦ Delete a submitted utility bill (DELETE on UtilityBills by bill_id)

Donors can perform the following operations:

◦ Browse all unpaid utility bills (GET on UtilityBills with is_paid filter)
◦ View details of a specific utility bill (GET on UtilityBills by bill_id)
◦ View their payment history (GET on Payments by donor_id)

Payments table (Users can perform the following operations:

◦ View payments with a GET operation (by bill_id and donor_id).
◦ Update payments with a PUT operation (change the donor_id or payment date if necessary).
◦ Delete payments with a DELETE operation (by payment_id).

- Implement a notification system to inform beneficiaries and donors of new bills or payments, as well as upcoming due dates.
- Allow users to connect their accounts to external payment systems like PayPal or Stripe for easier and more secure payments.
- Implement a feature to track partial payments and automatically notify users when a bill is fully paid off.
- Allow users to donate directly to a fund to help beneficiaries with their bills, rather than having to pay for individual bills.
- Implement a system to verify and authenticate users' identities, to prevent fraud and ensure that donations are going to those who truly need them.
- Create a dashboard for users to view and manage their bills and payments, as well as their donation history and impact.
- Allow users to leave ratings and reviews for each other, to help build trust and accountability within the community.

Tables:

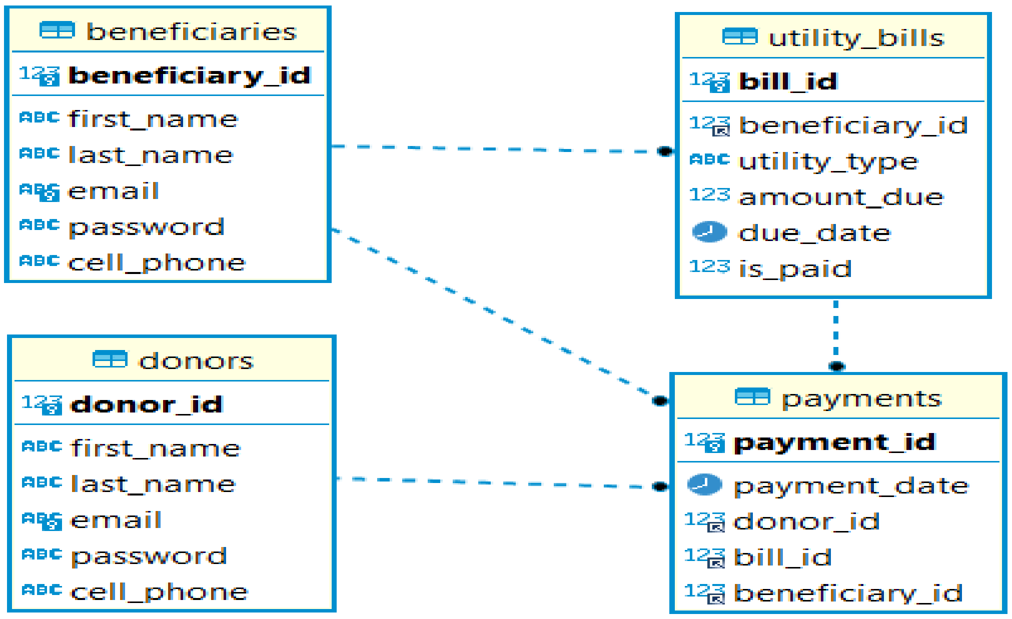
Beneficiary:
beneficiary_id (PRIMARY KEY),
first_name,
last_name,
email,
password,
cell_phone

Donor:
donor_id (PRIMARY KEY),
first_name,
last_name,
email,
password,
cell_phone

Utility Bill:
bill_id (PRIMARY KEY),
beneficiary_id (Foreign Key),
utility_type ENUM('water', 'electricity', 'natural_gas'),
amount_due,
due_date,
is_paid BOOLEAN NOT NULL DEFAULT 0

Payment:
payment_id (Primary Key),
payment_date,
donor_id (Foreign Key, refers to the donor),
bill_id (Foreign Key, refers to the utility bill),
beneficiary_id (Foreign Key, refers to the beneficiary

ERD



At least one 1-to-many relationship (PK/FK):

Between beneficiaries and utility_bills: A beneficiary can have multiple utility bills, but a utility bill belongs to one beneficiary. The beneficiary_id foreign key in the utility_bills table refers to the beneficiary_id primary key in the beneficiaries table.

A many-to-many relationship:

The relationship between beneficiaries and donors through the payments table; Each beneficiary can have multiple utility bills, and each utility bill can be paid by a single donor. A donor can pay for multiple utility bills belonging to different beneficiaries.

The payments table serves as a join table that connects beneficiaries and donors by linking the beneficiary_id from the beneficiaries table, the donor_id from the donors table, and the bill_id from the utility_bills table.

The many-to-many relationship is established between beneficiaries and donors because a beneficiary can be supported by multiple donors, and a donor can support multiple beneficiaries by paying their utility bills as there are three different categories of utility bills.

The payments table has foreign keys beneficiary_id, donor_id, and bill_id referring to the primary keys in the beneficiaries, donors, and utility_bills tables, respectively.