Description of Procedures

Overview

This app will connect those who want to donate to others with those who are looking for donations. It will use the following features to help donors and donatees to facilitate donations as well as allowing users to compete on who can be the most charitable.

Description

1. **Account creation:** users will be able to create accounts that they can use with the app. A user account will consist of a username, password, biography and phone number. The date of the account creation and the password will be stored in the database and be immutable. However, both the username and biography of a user may be changed by that user at any time.
2. **Editing of Accounts**: users will be able to edit or change details related to their account including their username, biography. They will be able to do this on the page of their account.
3. **Donation Request:** users will be able to make requests for donations by:
    1. **Donation Basket:** users will be able to create a donation basket for which they can search for certain items that they would like donated to them alongside the respective quantities of each item that they would like donated.
    2. **Posting Request:** users will be able to post their request so that it will be searchable by other users who may wish to respond to the request. The posting of their request will come in the form of a **Donation Page** which the user can name the page so that other users will more easily be able to find it.
    3. **Accept Donation:** when a donator gives to a donatee, the donatee will be able to accept or reject the individual items donated by the donator.
4. **Contributions:** users will be able to post that items that they wish to donate by selecting items that they want to donate and seeing the closest match for what they want to donate to what other users are asking for (excluding their own donation requests).
    1. **Contribution Basket:** users will be able to create a contribution basket for which they search for certain items that they would like to contribute and then add it to the basket along with the quantity of that item they would like to donate.
    2. **Search:** users will be presented with the option to contribute to specific donation pages (by page name) that they want to contribute to after creating their basket.
    3. **Submit Post:** users will be able to submit contribution baskets that the donatee may accept or reject (more specifically, the individual items within the contribution). When

users select a page that they want to contribute to, they can choose to donate everything in the basket, or a custom amount of each item they have in their basket (provided it does not exceed the amount in the basket). This will decrease the amount that a donatee needs (for the items listed on the donation page) as well as the amount available in the donor's basket. Users will also be able to *over* donate to a donation page i.e. exceeding the quantity of a specific item requested for donation.

5. **Account view:** users will be able to view their account as it is displayed to others as well as search for other accounts by their username or account id. This will consist of their username biography and phone number as well as their account id. Additionally, on the view of the account page of a user, their ***rank*** will be shown, for which lower ranks correspond to higher amounts of donation post acceptances and higher ranks correspond to lower amounts of donation post acceptances. If two (or more) users have the amount of donation post acceptances, then their rank is determined by which account was created later.
6. **Account History:** from the account page, users may view the history of an account in the form of what donation pages and what contributions have been created under that account. The contributions will list each entry in that contribution and whether that entry was accepted or not.
7. **Ranks:** Users will be be able to view the top donators of the app based on how many donation post acceptances they have, or who created their account later in the case that the respective count of donation post acceptances are equal.
8. **Rank Share:** on the page of a user's account, they will be able to share their rank on Whatsapp by clicking on their rank.
9. **Donation Pages:** donatees will be able to create their own pages detailing information on what they need, why people should donate, information about them (besides what's on their account) almost like a personal website. This will scale with larger donation programs to help even more people alongside with small scale donation programs. The donation page (as it stands) will be written in a subset of the HTML language (so that donatees cannot do things such as cross-site scripting or similar). This will be stored as an HTML file on the server, which can be accessed without installing the app, so that awareness can be spread without people necessarily needing to download the app.The donation page will have a basket of items that the donatee wants donated.

Tools

This app will use php for webservices, mysql as a DBMS and java with android studio for the front-end. It will store donation pages as HTML.

Business Rules

User Registration and Authentication

- Users must register with valid and verifiable information i.e valid phone number. Users' usernames must only consist of printable ASCII characters and their passwords may only consist of ASCII characters.
- Users cannot use the app without first creating an account.
- Users cannot edit the details of other users.
- Authentication should ensure the security of user accounts

Eligibility to Donate

- Users cannot donate illegal or stolen goods such as weapons or illegal substances.
- Users cannot donate goods which are illegal for the donatee to be in possession of e.g alcohol donation to a person not of legal age to drink alcohol .

Privacy and Security

- User privacy should be protected with sensitive information being handled in a secure fashion i.e connections to the server hosting the database will be done through HTTPS and not HTTP.
- Users' passwords will be encrypted so that it will be extremely difficult for one to use server-side exploits to gain access to a user's account.
- Personal Information is well-protected.

Compliance and Regulations

- The app must comply with relevant laws and regulations of or relating to the governing of donations and privacy

Terms of Service and User Agreement

- Users must agree with the terms and conditions of the app.

Development Of ERD

Initial ERD

Evaluation of Normalisation form

Since it is difficult to determine if a table is in Boyce-Codd normal form without using any formal definition, evaluation of the normalisation level of each table has been considered more

formally.

1NF:

All the tables are in first normal form already because:
- All values are already in ***atomic*** as they are all indivisible so cannot be split or decomposed into different attributes (reasonably).
- There are no ***repeating groups*** as each tuple in the table only contains information related to a single attribute.
- Each table has a ***primary key*** selected.

2NF

All tables are in 2NF as all tables are in 1NF and in each table, each non-prime attribute does not depend on any proper subset of a candidate key, i.e. there are no ***partial dependencies*** for any table.

3NF

To speak formally, a table is in 3NF if and only if the table is in 2NF and there does not exist a non-prime attribute of the table that is transitively dependent on the primary key, where a ***transitive dependency*** is defined to be a functional dependency where $X \to Z$ indirectly, through two other functional dependencies $X \to Y$ and $Y \to Z$ where it is not the case that $Y \to X$.
Equivalently, each non-prime attribute is determined only by an entire candidate key and never another non-prime attribute.
All tables in the database have no transitive dependencies as there does not exist any transitive dependency in any table within the database, there only exists dependencies between candidate keys and other attributes.
However, there is *almost* transitive dependencies. For example, `Accounts.username` determines all the attributes in the `Accounts` table but is not part of the primary key, however `Accounts.username` is actually a candidate key and hence by definition does not form a transitive dependency as `Accounts.username` $\to$ `Accounts.id`.

BCNF

A table is said to be in Boyce-Codd normal form if and only if for every functional dependency $X \to Y$, $X \to Y$ is trivial i.e $Y \subseteq X$ or X is a super key of the table.
Since no candidate keys in any table are overlapping, it follows that all tables are already in Boyce-Codd normal form the conditions for Boyce-Codd normal form naturally follow.

4NF

There are no multi-valued attribute in any table in the database. Hence all tables are already in 4th normal form

Issues

- The `Accounts` table has two candidate keys (although `Accounts.username` has semantic meaning and is non-numerical).
- The `Resources` table has two two candidate keys (although `Resources.name` has semantic meaning is non-numerical, so is not a good choice).
- `Contributions.accepted_at` is nullable, of course with null-values being considered a "billion dollar mistake" this perhaps isn't ideal.
- Each table related to `Accounts` is linked by two foreign keys a `from` foreign key and a `to` foreign key, so to speak. This makes the schema of the database more complicated, even if the relationships are distinct and disjoint i.e the relationships between `Accounts` and another table T cannot be put together without changing the meaning of the relationships and making the relationship between both tables less clear.
- There is technically still data redundancy in that surrogate keys are used when composite candidate keys already exist. This was done so as to prevent a difficult to understand database schema which would cause queries to be harder to understand and more "brittle" (i.e. if something changes in the database schema, it would be more problematic to correct all the queries that would be affected by the change).

Potential but avoided issues

- Referential integrity is ensured since all foreign keys are non-nullable.
- There are no multi-value attributes in the table.
- The database is fully normalised, or at least to the extent that the normalisation forms are covered.

ERD Second Version

To avoid null values for the `Contributions.accepted_at` attribute, this attribute was taken and moved to a new table, the `AcceptedContributions` table, which does not permit the nullation of its attribute `AcceptedContributions.accepted_at`. Hence no table has the possibility of null-values removing potential hard-to-track issues caused by treating a null value as if it were not null. The `AcceptedContributions` table will use the foreign linking it to the `Contributions` table as a primary key, as it would uniquely identify the table without being messy like with a composite key.

ERD Third Version

The database was extended and slightly modified, without changing the normalisation form, to incorporate new tables to keep track of pending contribution requests and better align to the functional requirements of the app as well as simplify the design and scalability of queries. As an example, `AcceptedContributions` was changed to `AcceptedContributionEntries` as it was asked for the raw quantity of items donated and not total posts accepted as listed in the `AssignmentDetails.pdf`, it now links to `ContributionEntries` instead in the same fashion that `AcceptedContributions` linked with `Contributions`.
Furthermore, `ContributionEntries.post_id` was renamed to `ContributionEntries.contribution_id`.
Furthermore, an *extremely* convenient attribute, `account_rank` along with `accepted_contributions`, was added to the `Accounts` table. Although technically a redundancy, as the rank of an account may be calculated without it, this table simplifies the design of the query to update the rank of a user in the table.
The decision was made to store donation pages as webpages, hence, they no longer store their content but instead reference a file stored within the server by their name. This does force the constraint that the name of a donation page is unique, but this constraint doesn't seem to pose any immediate or obvious issues, this removes the need for the `DonationPages.description` attribute.
The `PendingContributions` table was added, as it was initially overlooked. This makes designing the query for accepting donations much simpler than what would've been before.