

# MS805 Group Assignment

## Deliverable C — Design Flexibility Report

### Group: Thu 203

#### **1. Assumptions:**

While we were creating the Entity Relationship Diagram (ERD) for Share the Meal platform, there were some key assumptions that influenced how we made the database and how the different entities connect to each other. Our design has multiple tables, and we set up relationships with primary and foreign keys to make sure data management is smooth and remains intact.

##### Supporter

1. The supporter must create an account.
2. A supporter can volunteer for the campaign and can subscribe for it.
3. Supporters can make lots of donations, and every campaign can get donations too. This creates a many-to-many link between supporters and campaigns through the donations table, which keeps track of each donation that gets made.
4. For the ERD 1 we assume that the user logs in with their Google Account.

##### Campaign

1. For each campaign the donation amount is converted into number of meals which will be provided for that donation.
2. In ERD 1 we assume that each campaign can have a different status such as Active, Inactive and completed.

##### Challenges (**only for Proposal ERD 1**)

1. We assume that the user will give information regarding the challenge name, challenge description, end date as well as meal goal.
2. Each user can create multiple challenges. However, each challenge can be associated with only a single campaign.
3. Here as well, we assume that each campaign can have a different status such as Active, Inactive and completed.
4. Users can create these challenges only on the “Share the Meal” app if it is available on the website as well it would be convenient for the users. Here we assume that it can be done on the website as well.

## Donation

1. Donations can be done in different currencies, and each donation is linked to a particular currency.
2. Donations could be either one-time or part of subscription.
3. People can donate no matter where they are located.

## Beneficiary

1. Each beneficiary is linked to the campaign i.e. they are tied to the area where the campaign is happening. Beneficiaries connect to campaigns through a foreign key in the beneficiary table, letting each campaign support one or more beneficiaries.

## Campaign update

1. Campaigns have regular updates to keep supporters in loop about what's going on.
2. Campaigns can have many updates, and each update belongs to just one campaign, creating a one-to-many relationship between campaigns and their updates.

## Currency

1. Every location has its own currency, which has its own currency code, symbol, and exchange rate, there could be several locations using the same currency, so it ends up being a one-to-many relationship between currency and location.

## Campaign\_location

1. A campaign can be held in several locations, and each of those locations can host many campaigns. To make this work, we set up the campaign\_location table, which creates a many-to-many connection between campaigns and locations.

## Location

1. As each campaign is for a particular region, we plan to store the boundary coordinates of that region using geometry data type. (ERD 1).

## Image

1. Supporters, campaigns, or beneficiaries can have images, they could represent updates for campaigns, or profiles of supporters.
2. Since images can connect to different entities, the image table uses foreign keys for each type. This creates a one-to-many relationship for each kind.

## Subscription and Payment details

1. With subscription people can make regular donations to campaigns. Each one links to payment info, card numbers, expiration dates and billing addresses. Subscriptions and single donations are linked by foreign keys and one-to-many relationships to tell them apart.

2. Supporters might have one or more payment details saved in the payment\_detail table, linked by a foreign key to the supporter, which sets up a one-to-many connection between supporters and payment details.

## Volunteer

1. Supporters have the choice to volunteer for a campaign. So, we've got a separate volunteer table that sets up a one-to-many relationship between supporters and campaigns, since a supporter can help out in multiple campaigns, and each campaign can have several volunteers.

## Impact of Assumptions on ERD Design Decisions

1. The campaign\_location table for linking campaigns and locations, along with the direct connection of donations between supporters and campaigns, makes the design flexible enough to track how all these entities interact. These tables make it easier to search for donations by campaign, see supporter activities across campaigns, and understand the geographical spread of campaigns.
2. Making separate tables like payment\_detail, subscription, and campaign\_update helps organize data better and cuts down on redundancy. For example, payment\_detail keeps payment info, making it simpler to manage supporter payments and change their payment details without messing with donations or subscriptions.
3. The links between beneficiaries and campaigns, and how campaigns connect with locations, allows for flexibility in creating hierarchical relationships. It makes the model easier to adjust for adding more types of support and monitoring how campaigns expand across different areas.
4. With the currency table linked to donations and locations, this design makes currency management for international work much simpler. This is super important for handling different currencies without filling up multiple tables with the same currency info.

## 2.Comparative advantages and limitations:

Sr. No	Relational Database Designs	Advantages	Disadvantages
1	Proposal ERD 1	<ul style="list-style-type: none"> <li>focuses on making things easy for users with a detailed PAYMENT_METHODS table. It saves different</li> </ul>	<ul style="list-style-type: none"> <li>This design doesn't have a beneficiary entity, so it can't really record or look at the demographics or specific effects on the people who</li> </ul>

		<p>payment methods and lets user pick a default one, so user can handle flexible, recurring, and one-time payments without any hassle. The RECURRING_DONATIONS table takes care of regular donations automatically, which is super handy for donors who want to give periodically without having to do it manually every time.</p> <ul style="list-style-type: none"> <li>• This design has a CHALLENGE table that lets users set up fundraising challenges right in their campaigns. This feature is a great way to get people involved, getting supporters to create their own custom campaigns or challenges to boost donations. It really helps to build community and make people feel more personally invested. <b>(This feature is not available on the website, but it is there in the mobile application)</b></li> <li>• Even though it just keeps location data at the campaign level, this simple method makes it way easier to handle and not as heavy on resources. Tracking campaign locations gives enough info to look at how campaigns do in different areas without making things too complicated.</li> </ul>	<p>benefit from the campaigns. Because of this, it can't give much insight into how campaigns are impacting certain groups, like those who are more vulnerable.</p> <ul style="list-style-type: none"> <li>• Without an Image table, the platform can't show campaign visuals, user profiles, or pictures of beneficiaries. This kind of takes away from the visual engagement on the platform, which could otherwise make the campaigns more visible and help connect with supporters better.</li> <li>• This design doesn't include any way to manage volunteers, which makes it hard to keep track of volunteer roles in the campaigns. Because of that, there's not much chance for supporters to get involved in ways other than just giving money.</li> </ul> <p><b>(Design 2 covers these 3 limitations)</b></p>
2	Proposal ERD 2	<ul style="list-style-type: none"> <li>• It introduces a beneficiary table, allowing for the</li> </ul>	<ul style="list-style-type: none"> <li>• This design takes care of payments using a payment</li> </ul>

		<p>tracking of individuals or groups who benefit directly from each campaign. This addition makes it easier to assess the impact of campaigns on specific demographics, providing valuable insights into how many children, elderly, or vulnerable people are supported. (<b>unlike Design 1</b>)</p> <ul style="list-style-type: none"> <li>• The <b>image</b> table lets us store and show off campaign visuals, which really boosts engagement and transparency. Plus, keeping track of location info for supporters, campaigns, and beneficiaries gives us a nice dataset that helps analyse how far campaigns reach and where the supporters are.</li> <li>• With the new volunteer table, this Design lets us keep an eye on volunteers, making it easier for supporters to jump in in different roles. This adds more ways for people to help out with campaigns, making the platform pretty flexible.</li> <li>• Because of the dedicated <b>currency</b> table, this design is ready for international help, so donations and payments can get processed in different currencies depending on where the supporter and campaign are. This is super helpful for platforms with</li> </ul>	<p>table that connects with donations, but it doesn't keep any user-specific payment preferences or default payment choices. While it does a good job at tracking transactions, this setup doesn't make recurring payments as easy as <b>Design 1</b> does, since it relies on a subscription table for manual tracking instead of automatic payments.</p> <ul style="list-style-type: none"> <li>• With its wider structure that includes beneficiaries, volunteers, and media, <b>Design 2 ends up being trickier</b> to set up and manage. It might need more resources and storage, plus extra logic to keep the data consistent across the many linked tables.</li> </ul>
--	--	--	--

		users all over the world, as it makes cross-border transactions and currency swaps a lot easier.	
3	Proposal ERD 3 - Generated by Copilot	<ul style="list-style-type: none"> <li>• The AI did a good job of making the ERD simpler by zeroing in on the key parts, which is good for times when you need to roll things out quickly instead of having every single feature.</li> <li>• The design brings in some fresh and often ignored ideas like PartnerOrganization and Event, which really help boost value through partnerships and campaigns tied to events. This makes it a great fit for bigger, teamwork-focused campaigns that depend on working with other groups.</li> <li>• The Feedback table lets supporters share their thoughts, giving campaign organizers a peek into what users feel and how engaged they are. This part can really help with ongoing improvement and builds a sense of community around the campaigns.</li> </ul>	<ul style="list-style-type: none"> <li>• The AI missed out on a bunch of features that are usually part of donation and campaign platforms, like having a way to make recurring payments and keeping track of beneficiaries in detail, which are super important for the success of campaigns in the long run and measuring their impact.<b>(included in Design 1 and 2)</b></li> <li>• While the design does include basic campaign and user images, it lacks a direct link to beneficiary demographics. This limits the platform's ability to measure and report on the specific impact of campaigns on different beneficiary groups, which could be crucial for impact analysis.</li> <li>• Although this design includes latitude and longitude data in the Location table, it does not provide multi-level location tracking (e.g., for supporters and beneficiaries) <b>as in Design 2</b>. This limits geographical insights into supporter demographics and campaign reach.</li> </ul>

### 3.Flexibility:

#### (a) Prior to Database Rollout

Before things get rolling, it's pretty simple and not too costly to add or change tables, attributes, and relationships. Since the database isn't live yet, developers can test changes over and over to ensure everything's working right and the data remains safe, all without bugging any users. If there's any mismatch in naming conventions (like "User" vs. "Supporter"), fixing those is a breeze at this point. For example, making sure "location" is represented the same way or merging donation-related attributes is totally doable. Before launch, it's easier to set up consistency rules (like standards for currency fields) and to enforce business logic needs.

The workload is moderate, since it needs some thoughtful design and testing, but it doesn't really disrupt anything. Standardizing things across different designs, especially with modular stuff, might just take a few development sprints.

#### **(b) After Database is Live and in Use**

Just making those changes once the system is up and running. Existing data needs to be mapped and moved carefully to match any new setup, which involves a ton of detailed work in transforming, validating, and testing data to make sure nothing gets lost or messed up.

A live setup needs some downtime for schema changes, especially for the bigger ones. This could mean planning maintenance times, letting users know, and having rollback plans ready just in case.

Switching to a new design needs to make sure it still works with any front-end or third-party apps that depend on the current setup. For example, if "Campaign" gets updated to handle more entities, you'd have to change API endpoints and other connections.

Any app logic, triggers, or stored procedures that work with the database will need updates to play nice with the new schema.

In a live system, doing extensive testing is super important to ensure that everything runs smoothly after the changes. This includes checking for performance issues since changing the schema can mess with how queries get optimized.

Changes impacting user interfaces or workflows might mean users need retraining and the documentation has to be updated.

The workload is high, as making these live changes means you need careful planning, extra testing, and validation to keep data intact and minimize disruptions. The transition could take several cycles, especially with larger changes.

### **4.Sociotechnical considerations:**

When looking at the social and technical factors in designing and launching a system, there's a bunch of important stuff to think about.

- **Stakeholders and Sensemaking:** Making sure all stakeholders—like donors, campaign organizers, and system admins—understand clearly what the system's for, how it works, and how to read the data inside it.

If stakeholders see things differently, it can create different expectations and not-so-great usage. Keeping them in the loop with regular and clear communication (through meetings, documents, trainings) helps everyone stay on the same page.

- **Organizational Silos and Boundary Spanning:** Organizational silos are when teams work by themselves with little chat happening between them, which can really mess up how well systems work together and how smoothly users feel. Boundary spanning is about building connections between these silos to get better collaboration.

This is super important if different teams are handling separate bits of the system (like fundraising versus data management). Having boundary-spanning roles or using things like shared dashboards or cross-functional teams can help keep things consistent and understood across the silos.

- **Upgrading with Care:** For hassle free transaction a careful upgradation is must. Planning upgrades, rolling them out bit by bit, and doing thorough testing can help avoid interruptions. Telling users about changes ahead of time lets them adjust, especially when it comes to live data or important donation processes that might be affected.
- **Corroboration and Entity Resolution:** Having accurate data is super important for building donor trust, reporting on campaigns, and keeping the platform reliable. Making sure donor identities, campaign records, and donation histories line up across the system helps avoid duplicates and wrong reports that could hurt credibility. Putting in strong data cleaning and checking processes will help keep records consistent and trustworthy.

**Conclusion:** To sum things up, creating a system for Share the Meal means putting the spotlight on key things like talking with stakeholders, making thoughtful upgrades, and keeping data accurate to boost how well it works and how much people can trust it. Some other important stuff, like crossing organizational boundaries and dealing with time-sensitive features, does not affect the mission. If everything's running smoothly, the platform can still do a great job and really make a difference in the fight against hunger.