FINAL PROJECT PROPOSAL

Project Participants: Justin Turner

Title: Public Health Nonprofit Program Funding

Executive Summary:

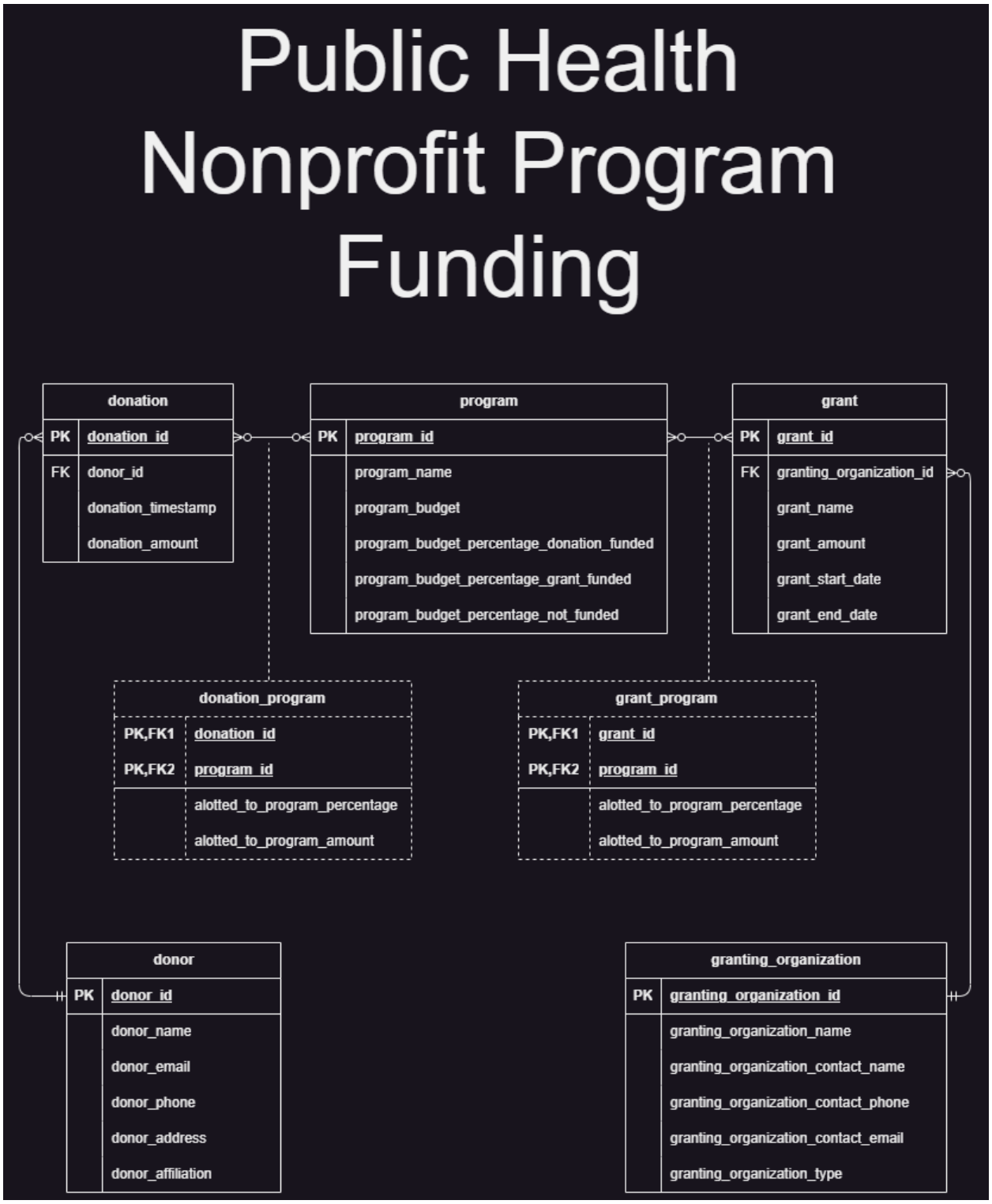
Given a fictional nonprofit organization focusing on public health programs, I will create a database architecture representing various specific programs run by that organization, with records on how they are funded, as described below:

- The **Program** table will contain information such as program name, program budget, percentage of budget relying on donations, and the percentage of the budget relying on grants.
- The **Donor** table will contain information such as the donor's name, contact info, and organizational affiliation.
- The **Donation** table will contain information such as the date the donation was made, the monetary amount, and a reference to the donor making the donation.
- The **Grant** table will contain information such as the monetary amount given, the length of time for which the grant is active, and the name of the grant.
- The **Granting Organization** table will contain information such as the name of the organization, whether that organization is federal or private, and contact information for that organization.

Some information about relationships between the entities:

- The **Donor** and **Donation** tables will have an **optional one-to-many relationship**. I will reference entries in the donors table within the donations table using the donor_id as a foreign key.
- The **Program** and **Donation** tables will have an **optional many-to-many relationship** (each program will have many donations associated with it, and each donation may have many programs it was intended to help finance, but donations can also be made to the nonprofit without specifying a program). I will use a **join table** to link donations and programs.
- The **Program** and **Grant** table will have an **optional many-to-many relationship**, and also use a **join table**.
- Finally, the **Granting Organization** and **Grant** table will have an **optional one-to-many relationship** (one organization may provide one or more grants, but records can exist), and the grants table will reference entries in the granting organizations table using the granting_organization_id as a foreign key.

Entity Relationship Diagram:



Description of endpoints/features:

/program

**note: only one parameter would be included in any given request or the application will return a 400 response code with an additional message to the user specifying this requirement. All parameters sent to* GET /program *would take the form of query strings.*

Verb	Parameters	Response	Request Body
GET	*none*	A list of all programs and their associated data.	*none*
	donor_id	A list of all programs to which a specific donor has donated.	
	granting_organization_id	A list of all programs with grant funding from a specific organization.	
	donation_id	A list of programs to which a specific donation applies.	
	grant_id	A list of programs to which a specific grant applies.	
POST	*none*	The created program and a 201 status code	JSON representing the program entity

/program/*program_id*

Verb	Response	Request Body
GET	The program entity with the id specified, or a 404 status code if the program id is not found	*none*
PUT	The updated program entity and a 200 status code	JSON containing any fields intended to be updated

/donor

Verb	Response	Request Body
GET	A list of all donors and their associated data	*none*
POST	The created donor entity and a 201 status code	JSON representing the donor entity

/donor/*donor_id*

Verb	Response	Request Body
GET	A specific donor entity, or a 404 status code if the donor id is not found	*none*
PUT	The updated donor entity and a 200 status code	JSON containing any fields intended to be updated
DELETE	A 204 (no content) status code, if successful. If a donor has any associated donations, a 403 (forbidden) status code will be returned instead.	*none*

/donation

**note: only one parameter would be included in any given* GET *request or the server will return a 400 response code with an additional message to the user specifying this requirement. All parameters sent to* GET /donation or POST /donation *would take the form of query strings.*

Verb	Parameters	Response	Request Body	Notes
GET	*none*	A list of all donations and their associated data.	*none*	
	donor_id (optional)	A list of all donations made by a specific donor.		
	program_id (optional)	A list of all donations linked to a specific program.		
POST	program_id (optional)	The created donation and a 201 status code	JSON representing the program entity. The optional program_id parameter would associate the donation with the program with the given id at the full amount.	For any donations that are intended to be split across multiple programs, see PUT requests in the next section.

/donation/*donation_id*

Verb	Response	Request Body	Notes
GET	A specific donation entity, or a 404 status code if the donation id is not found	*none*	
PUT	The updated donation entity and a 200 status code. If including the programAllotments field, also return a list of entities representing the corresponding entries in the donation_program join table.	JSON containing any fields intended to be updated.	If the intention of the PUT request is to specify how a donation is allotted to multiple programs, a field named “programAllottments” should be included in the JSON, and would take the form of a JSON object where keys would be program ids and values would be a monetary amount. Donations to a single program should still be sent in this format: the programAllottments JSON object would just contain a single entry.

/granting_organization

Verb	Response	Request Body
GET	A list of all granting organizations and their associated data	*none*
POST	The created granting organization entity and a 201 status code	JSON representing the granting organization entity

/granting_organization/*granting_organization_id*

Verb	Response	Request Body
GET	A specific granting organization entity, or a 404 status code if the granting organization id is not found	*none*
PUT	The updated granting organization entity and a 200 status code	JSON containing any fields intended to be updated
DELETE	A 204 (no content) status code, if successful. If a granting organization has any associated grants, a 403 (forbidden) status code will be returned instead.	*none*

/grant

**note: only one parameter would be included in any given* GET *request or the server will return a 400 response code with an additional message to the user specifying this requirement. All parameters sent to* GET /grant or POST /grant *would take the form of query strings.*

Verb	Parameters	Response	Request Body	Notes
GET	*none*	A list of all grants and their associated data.	*none*	
	granting_organization_id (optional)	A list of all grants made by a specific granting organization.		
	program_id (optional)	A list of all grants linked to a specific program.		
POST	program_id (optional)	The created grant and a 201 status code	JSON representing the program entity. The optional program_id parameter would associate the grant with the program with the given id in full.	For any grants that are intended to be split across multiple programs, see PUT requests in the next section.

/grant/*grant_id*

Verb	Response	Request Body	Notes
GET	A specific grant entity, or a 404 status code if the grant id is not found	*none*	
PUT	The updated grant entity and a 200 status code. If including the programAllotments field, also return a list of entities representing the corresponding entries in the grant_program join table.	JSON containing any fields intended to be updated.	If the intention of the PUT request is to specify how a grant is allotted to multiple programs, a field named “programAllottments” should be included in the JSON, and would take the form of a JSON object where keys would be program ids and values would be a monetary amount. Grants applying to a single program should still be sent in this format: the programAllottments JSON object would just contain a single entry.
DELETE	A 204 (no content) status code if the grant entity was deleted, otherwise a 404 status code if no grant associated with the provided id is found.	*none*	If a grant is deleted, corresponding entries in the grant_program join table should also be deleted.

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

- Additional query string parameters on the GET /donation and GET /grant endpoints allowing users to filter responses by timestamp
- Full **RAML** documentation of all endpoints.
- Using Spring Security to authenticate users and limit access to specific endpoints based on various criteria.