Test Creator System Design

Nathan Gurrin-Smith

May 18, 2023

1 Introduction

This is a program that will be used to generate practice tests from a list of questions. It aims to help improve learning by encouraging frequent, interleaved testing.

2 Use Cases

- Login,
- Add, view, edit, and delete presets,
- Add, view, edit, and delete questions,
- add, view, edit, and delete categories,
- generate practice tests

3 Technology

The framework being used is PERN.

3.1 Data

Naming Conventions

We will use the following naming conventions for our data model

- Tables and columns will be in PascalCase.
- Tables and columns will be singular.
- Junction tables will be named JunctionTable1Table2

Note: PostgreSQL automatically forces lowercase internally, but we'll always reference by PascalCase.

Note: Because of the above note, we must reference all columns with lowercase in our javascript files.

Data Model

We will have the following data model:

UserAccount		
UserAccountID	Username	Password

ClientAuthentication				
ClientID	LastLogin	RefreshToken	RefreshTokenExpiry	UserAccountID (

Preset					
PresetID	Name	Preamble	Sep	Postamble	UserAccountID(FK)

Collection	
CollectionID	Name

SubCollection		
SubCollectionID	Name	CollectionID(FK)

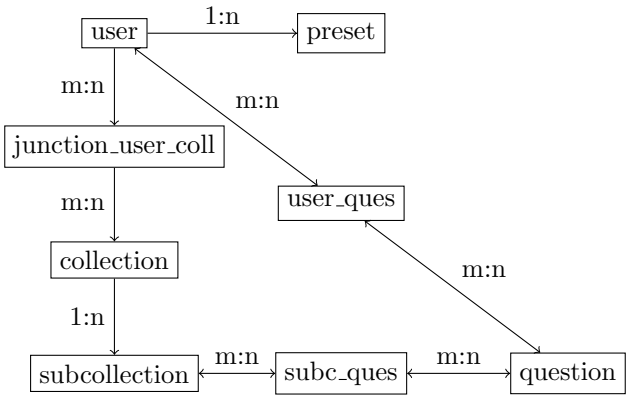
Question			
QuestionID	Name	Content	Source

JunctionUserAccountCollection	
UserAccountID(FK)	CollectionID(FK)

JunctionUserAccountQuestion		
UserAccountID(FK)	QuestionID(FK)	LastReviewed

JunctionSubCollectionQuestion	
SubCollectionID(FK)	QuestionID(FK)

Which are related via,



Orphans: Some of the properties above require a parent.

- A Collection requires at least one parent UserAccount,

- A SubCollection requires a parent Collection,
- A Question requires a parent UserAccount and a parent SubCollection
- A Preset requires a parent UserAccount

Thus, when we delete a parent, we not only need to delete entries in the junction table, but also the entries in the child tables since, most frequently, the foreign keys will be stored in the junction table and not the child table. The ones that need this, that is, the ones that aren't handled by ON DELETE CASCADE, are orphaned collections and orphaned questions. So, we need to handle orphans when we delete:

- Users (questions and collections)
- Subcollections (questions)

3.2 Backend

Routes

We are using the `/api/` style.

- All user related routes will be at /api/user/
- All preset related routes will be at /api/preset/
- All collection related routes will be at /api/collection/
- All subcollection related routes will be at /api/subcollection/
- All question related routes will be at /api/question/

Other Modules

Here are the other modules we have,

- dbClean.js: as mentioned above, we need to ensure orphaned entities are deleted, which this module handles.

Client