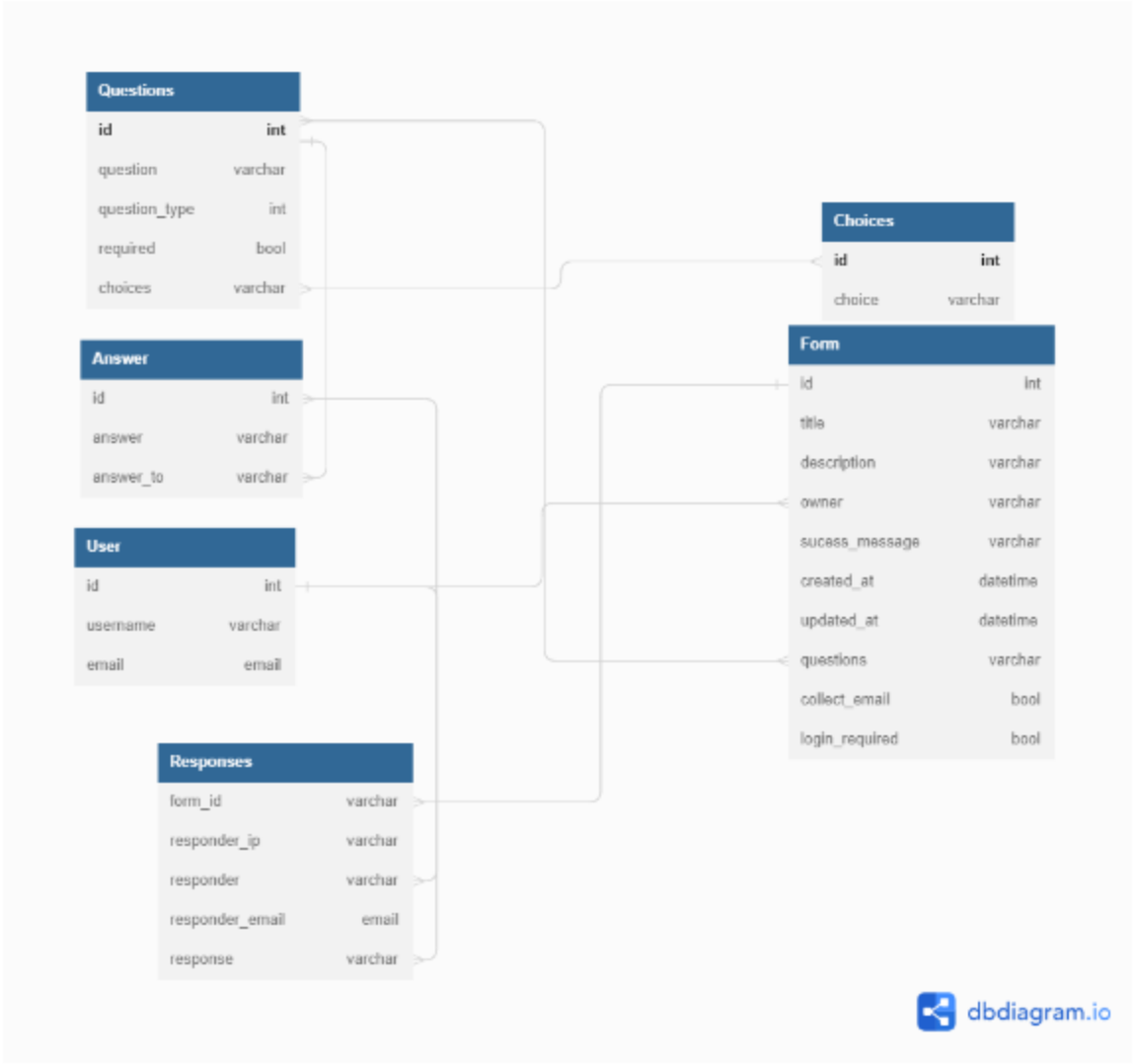
Design Specification

1. Database : SQL-Based database (In this demo I have used SQLite database because its' a default database in django and also easily portable between systems. However in a real world scenario I would have used **MYSQL or postgres databases.**)



2. API and Views:

- **def home(request):**

  It returns all the forms of the logged in user in My forms sections and all the available forms across all the user in the Other available forms section.
  If the user is not logged in My forms section will be empty.

- **def show_form(request, key):**
  Takes the key variable and displayed the Form from the Form table having that key as the primary key.

- **def save_response(request, key):**
  Takes all the responses for each of the questions and saves them into the database and further redirects the user to the home page showing the success message.

- **download_excel(request, key):**
  This view takes a key as an argument and downloads all the responses belonging to that particular form as a .csv file and downloads it.

- **def save_survey(request):**
  This view saves the form structure and questions created by the user into the database. So that it can be available for the users to fill when the for is published.

2. Caching Response using javascript:

I used javascript LocalStorage to cache the response for when the form is shared to the user to fill it due to less time constraints. Whereas in real world scenarios **redis** database can be used.
Later if the user reloads the page the response stored as cache is used to auto fill the form and thus the user doesn't have to refill it.

Pro/cons Analysis:

For selection of database using I was having two options of either using SQL based or No SQL based database. Here a NOSQL based database may be a good fit but it could not be scaled well over the increasing number of forms and responses. Whereas SQL databases are fast and vertically scalable.

So I finally ended up selecting a SQL-based database, and here the default database can be the best choice for making demo apps. But on the deployment or during scaling we can definitely model to SQL or postgres databases without any overhaul changes in the codebase. I just have to connect the new database into setting.py file.

Advantages:

- I have made Question and choices into different tables and connected it by foreign key so that we can have a variable number of choices while creating MCQs and no choices for que types such as True/ False, time date etc.

Limitations:

- I can use the redis database for caching which will be more efficient.

Pledge:

I confirm that all of the information provided in this submission is my own work and, to the best of my knowledge, complete and accurate.

Name: Vishal Singh
Date: 26/10/22