# Database Re-Design

Christ Mboungou 04/02/20222

## Overview

There are many relational database systems to choose from for my capstone project. I decided to use PostgreSQL as my database. PostgreSQL is one of the most popular open-source relational databases systems globally, with a robust feature set related to performance, security, programming extensions, and many more. The main reason I picked PostgreSQL is that it's free and its object-oriented database features. I anticipate that the timeline feature will need complex querying and relationships, and I think PostgreSQL will be a perfect fit for the MVP.

I will not use an API for this MVP. I will be using the MVT structure. Model View Template (MVT) is a software design pattern for developing web applications. Even though I will not be writing SQL queries, I will use Django's database mapper to handle database calls. Django will automatically give me a database-abstraction API that lets me create, retrieve, update and delete objects without writing SQL queries. If I need to write SQL queries that are too complex for Django's database-mapper to handle, I fall back on writing SQL by hand.

## Database Structure

Our database will be normalized. This process will allow us to efficiently organize, eliminate redundancy and ensure data dependency. This will help us reduce the amount of space our database consumes and ensure that data is logically stored. In our application, we will use the Third normal form principle.

## Data specifications

Table name: **Users**

Django handles user registration and authentication for me automatically. We had to import the RegisterPage view from Django and set up a user creationForm imported from Django.contrib.auth.forms.

| column_name | data_type | Constraints |
|---|---|---|
| id | integer | Primary Key |
| password | character varying | |
| last_login | timestamp with time zone | |
| is_superuser | boolean | |
| username | character varying | |
| first_name | character varying | |
| last_name | character varying | |
| email | character varying | |
| is_active | boolean | |
| date_joined | timestamp with time zone | |

## Table name: Goal

The goal table will hold all the necessary data needed for goal creation. This table will have a Foreign Key user_id which will associate users with a goal whenever a goal is created.

| column_name | data_type |
|---|---|
| id | Primary key |
| title | character varying |
| user_id | integer |
| completed | boolean |
| description | text |
| end_date | timestamp with time zone |
| start_date | timestamp with time zone |
| type | text |
| category | character varying |

## JSON Structure:

```
{
    "id": autogen PK,
    "user": int,
    "title": string,
    "description": string,
    "type": string,
    "category": string,
    "completed": Boolean,
    "start_date": datetime,
    "end_date": datetime,
    "gpa": null
    "sat": null
}
```

## Table name: SAT

The SAT table will have a one-to-one relationship with the goal table. A goal can only have one sat record associated with it. Goal_id is the foreign key associating SAT and Goal table.

| column_name | data_type |
|---|---|
| id | Primary key |
| practice_test_score | double precision |
| private_tutor_time | double precision |
| have_a_strategy | boolean |
| goal_id | Foreign key |

## JSON Structure:

```
{
    "id": autogen PK,
    "user": int,
    "title": string,
    "description": string,
    "type": string,
    "category": string,
    "completed": Boolean,
    "start_date": datetime,
    "end_date": datetime,
    "sat": {
        "id": autogen pk,
        "practice_test_score": float,
        "private_tutor_time": float,
        "have_a_strategy": Boolean,
        "goal": int
    },
    "gpa": null
}
```

## Table name: GPA

The GPA table will have a one-to-one relationship with the goal table. A goal can only have one GPA record associated with it. Goal_id is the foreign key associating GPA and Goal table.

| column_name | data_type |
|---|---|
| id | Primary key |
| current_gpa | double precision |
| library_hours | double precision |
| friends_with_high_gpa | double precision |
| office_hours | double precision |
| goal_id | Foreign key |

## JSON Structure:

```
{
    "id": autogen PK,
    "user": int,
    "title": string,
    "description": string,
    "type": string,
    "category": string,
    "completed": Boolean,
    "start_date": datetime,
    "end_date": datetime,
    "gpa": {
        "id": autogen pk,
        "current_gpa": float,
        "library_hours": float,
        "friends_with_high_gpa": int,
        "office_hours": float,
        "goal": int
    }
    "sat": null
}
```

# Purpose, implementation, and interactions

## User

## Purpose:

The user table is critically important for numerous reasons. This table will have separate user information and give the application a sense of privacy. Without user records, there won't be a login or logout functionality.

Implementation:

This table handles user registrations as well as user login and logout. Upon login, this table will check if the user exists and authenticate.

Interaction:

When anyone tries to access the application, they will be directed to the landing page, which prompts them to either login or register an account. Both of these transactions will be handled by this table.

## Goal

Purpose:

The goal table will enable users to create a goal, and users will be required to select a category. A new record will also be created in either the SAT or GPA table, depending on what category the user selected. This table aims to house the user's basic goal information.

Implementation:

The goal table has a relationship with the user table, so the user must exist to successfully create a goal. Whenever a goal is created, it will automatically create either an SAT or a GPA based on the category selected, so users have to choose a category of their choice and fill out the required fields.

interaction:

Users must fill out the required fields and select a category to successfully create a goal when creating a goal.

## SAT

Purpose:

The SAT's purpose is to house SAT-type goals. This table allows us to enforce each goal to have only one SAT record.

Implementation:

The SAT records get created, updated, or deleted whenever the goal is created or updated by users.

interaction:

Users must fill out the required fields if the "SAT" is selected as a category.

## GPA

Purpose:

The GPA's purpose is to house GPA-type goals. This table allows us to enforce each goal to have only one GPA record.

Implementation:

The GPA records get created, updated, or deleted whenever the goal is created or updated by users.

interaction:

Users must fill out the required fields if "GPA" is selected as a category.

# Stretch Feature

The future development of this project will become much easier once the MVP is complete. We plan to add more categories to the list of goals so users can add goal categories such as SAT, GPA, College Application, and Personal Growth.

## College Application

| column_name | data_type |
|---|---|
| id | Primary key |
| How_many_deadlines | integer precision |
| Gather_financial_info | character varying |
| Submit_test_scores | boolean |
| Complete_essay | boolean |
| Double_check_work | boolean |
| Submit_application | boolean |
| goal_id | Foreign key |

### JSON Structure:

```
{
    "id": autogen PK,
    "user": int,
    "title": string,
    "description": string,
    "type": string,
    "category": string,
    "completed": Boolean,
    "start_date": datetime,
    "end_date": datetime,
    "college_application": {
        "id": autogen pk,
        "How_many_deadlines ": int,
        "Gather_financial_info": string,
        "Submit_test_scores": boolean,
        "Complete_essay": boolean,
        "Double_check_work": boolean,
```

```
            "Submit_application": boolean,
            "goal": int
        }
        "sat": null,
        "gpa": null,
    }
```

Purpose:

The College application category will work like SAT and GPA. This category's purpose is to serve different users.

Implementation:

College application records get created, updated, or deleted whenever the goal is created or updated by users.

interaction:

Users must fill out the required fields if "College Application" is selected as a category.

## Personal Growth

| column_name | data_type |
|---|---|
| id | Primary key |
| Make_gratitude_list | character varying |
| Check_financial_wellness | character varying |
| List_your_needs_for_the_day | character varying |
| Exercise_gym_hours | double precision |
| Unplug_hours | double precision |
| Recap_the_day | character varying |
| goal_id | Foreign key |

## JSON Structure:

```
{
    "id": autogen PK,
    "user": int,
    "title": string,
    "description": string,
    "type": string,
    "category": string,
    "completed": Boolean,
    "start_date": datetime,
    "end_date": datetime,
    "college_application": {
        "id": autogen pk,
        "Make_gratitude_list": string,
        "Check_financial_wellness": string,
        "List_your_needs_for_the_day": string,
        "Exercise_gym_hours": float,
        "Unplug_hours": float,
        "Recap_the_day": string,
        "goal": int
    }
    "sat": null,
    "gpa": null,
}
```

Purpose:

The personal growth category will work like SAT and GPA. This category's purpose is to serve different users.

Implementation:

Personal growth records get created, updated, or deleted whenever the goal is created or updated by users.

interaction:

Users must fill out the required fields if "Personal growth" is selected as a category.