Service concept

We are going to implement a service in python to be able to track habits. This will be a rest api service on django framework (with sqlite database). This document will have 3 sections: a description of the business logic, a description of the interaction methods with service, and a description of the necessary data structures for this service.

Business logic (entities)

We decided to create a rest api service because we want to enable anyone to use the functionality of our service. In the future, any service can use our business logic to improve user retention and business performance as a result.

In our service, you can register a **user**, for this you need to specify an email.
In addition, you can register any **habit** in our service. A habit consists of a name, a description, and a frequency type (daily, weekly, monthly, yearly)
In our service, you can create a **habit track**. Habit track consists of a link to the user, habits, longest streak. current streak.
We also create a **habit log** in our service. It consists of a link to a habit track and the time when the task was completed.
In addition, we decided to add an **achievement** entity to our project. Achievement consists of a name, description, link to a habit and rules of assign achievement to users.
The last entity in our service is **user_achievements.** User achievements consists of a link to the user and achievement.

Description of the interaction methods

In this section, we will describe the API methods of interaction with our service.

1) Create user - POST. Create user in system

```
/create_user {email: 'example@gmail.com'}
```

2) Get user - GET. Return user habits and streaks.

```
/get_user?email=example@gmail.com
```

3) Create habit - POST. Create habit in system

```
/create_habit {title: 'brush teeth', description: ‘description
example’, period: ‘DAY’}
```

4) Get habits - GET. Get all habits in the system.

```
/get_habits
```

5) Register user for habit - POST. Start tracking habit for user.

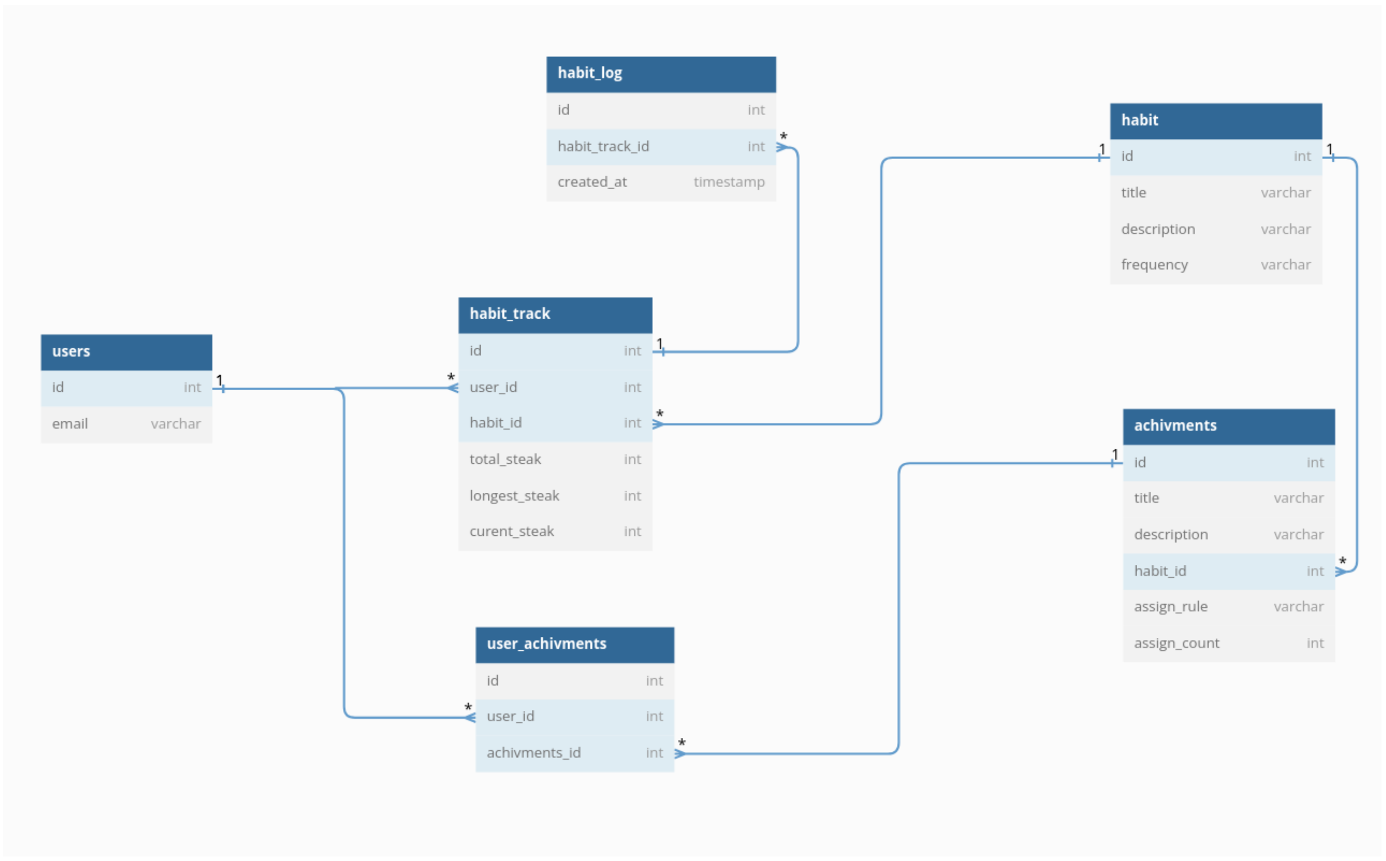
```
/register_habit {user: ‘example@gmail.com’, habit: 1}
```

6) Create new habit event - POST. User checked-off

```
/register_habit {user: ‘example@gmail.com’, habit: 1}
```

Description of the interaction methods

In this diagram, we have marked all the classes necessary for work, as well as the relationships between them.



habit_log
id int
habit_track_id int
created_at timestamp
habit
id int
title varchar
description varchar
frequency varchar
users
id int
email varchar
habit_track
id int
user_id int
habit_id int
total_steak int
longest_steak int
curent_steak int
achivments
id int
title varchar
description varchar
habit_id int
assign_rule varchar
assign_count int
user_achivments
id int
user_id int
achivments_id int
1
*