

Habit App Backend

Django, SQLite



Table of Contents

1. Framework selection
2. Stages of work
3. Business Logic
4. Testing
5. Outputs
6. Conclusion

Framework Selection

This project is developed using Django Web Framework for following purposes

- Django is a Python framework that makes it easier to create web sites using Python.
- Django was created with the goal of creating a framework that would allow developers to build web applications in less time.

SQLite is used as it is transactional, self-contained and needs minimal configuration.

Stages of work

Database Design

- Design Database Structure
- Design entity models and develop class relationships.
- Document the business logic.

API Development

- Develop endpoints using Django framework.
- Store records in SQLite database using SQLite3 library.

Testing and Debugging

- Create test fixtures and perform Unit Testing using the same.
- Find for any bugs and debug if any.

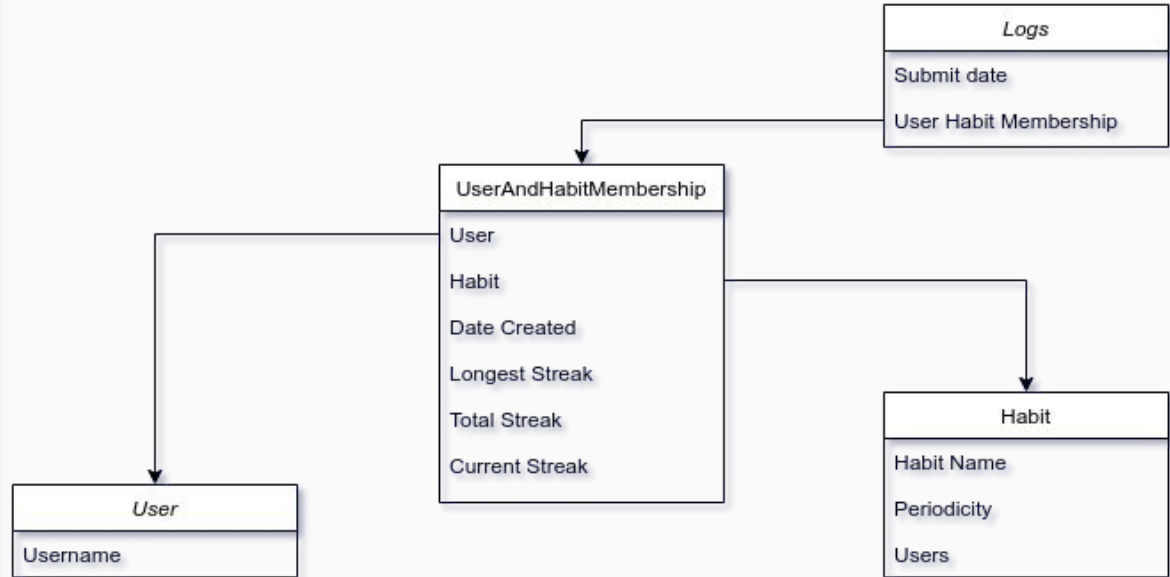
Business Logic

- A user registers to the project using an unique username.
- User can register to any pre-existing habit or they can create their own new habit. Habit contains information like habit name, periodicity and users linked to the habit.
- Users can track their habits streak as well as check the habit rating.
- All records are stored as a log as soon as any task is completed.

Class Diagram

The different entities in our project are as follows:

- User
- Habit
- UserAndHabitMembership
- Logs



Testing

```
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Habit Creation Done
User Creation Done
Add Habit to User Done
Checking off by user done
Checking off by user done
Checking off by user done
get_user Done
get_habit_score Done
Fixture Testing Completed
.
-----
Ran 1 test in 0.100s

OK
Destroying test database for alias 'default'...
```

Postman Outputs

http://127.0.0.1:8000/habits/create_user?username=test001

Save

POST

http://127.0.0.1:8000/habits/create_user?username=test001

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> username	test001			
Key	Value	Description		

Body

Cookies

Headers (8)

Test Results

200 OK 121 ms 288 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "user_id": 2
3 }
```

http://127.0.0.1:8000/habits/create_habit

Save

POST

http://127.0.0.1:8000/habits/create_habit

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> habit_name	Yoga			
<input checked="" type="checkbox"/> periodicity	DAY			
Key	Value	Description		

Body

Cookies

Headers (8)

Test Results

200 OK 149 ms 289 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "habit_id": 1
3 }
```


Postman Outputs

http://127.0.0.1:8000/habits/register_user_for_habit

POST ▼ http://127.0.0.1:8000/habits/register_user_for_habit

Params Authorization Headers (8) Body Pre-request Script Tests Settings

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> user_id	1
<input checked="" type="checkbox"/> habit_id	1
Key	Value

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ▼

```
1
2
3  "reg_habit_id": 1
```

POST ▼ http://127.0.0.1:8000/habits/new_habit_action

Params Authorization Headers (8) Body Pre-request Script Tests Settings

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

KEY	VALUE	DESC
<input checked="" type="checkbox"/> user_id	1	
<input checked="" type="checkbox"/> habit_id	1	
<input checked="" type="checkbox"/> submit_date	2022-11-01	
Key	Value	Desc

Body Cookies Headers (8) Test Results 200 C

Pretty Raw Preview Visualize JSON ▼

```
1
2  "habit_name": "Yoga",
3  "total_streak": 1,
4  "longest_streak": 1,
5  "current_streak": 1
6
```

Conclusion

- The project is developed using Django and SQLite libraries.
- Future scope includes visualisation feature for analytics module, making the login process more secure and so on.
- Django applications like these can be deployed on the internet very easily.

Thanks!

- Ilya Vasiliev

