Overview

You have been assigned with creating the primary REST service for a "Subscription as a Service" startup . Your REST service will be used by companies that will define subscriptions plans to which the users of the platform can subscribe to.

Instructions

General Instructions

The submission will be evaluated for:

- Correctness
- Code Readability: Please write well formatted, readable code with appropriate variable naming

Technical Instructions

- You MUST implement the API using Nodejs. You can use any framework on top of Node.
- You MUST use an RDBMS to store data (one of Postgres or MySQL).

Submission

- Share the assignment as a link to git repo
- The git repo needs to track the commits you made as you solved the challenge. This is so that we can see how your code evolved.

API Service

The primary aspect of this programming challenge is to implement the following two APIs:

- `/user`
- `/subscription`

The details of each of these APIs are as follows:

`/user`

This is a simple CRUD API that adds a user to DB.

`PUT /user/`

- creates a user with specified username in the DB.

Sample Input: `PUT /user/jay`
Required Output: Just a HTTP status: 200 on success, other appropriate code for failures

---

`GET /user/< username>`

Sample Input: `GET /user/jay`
Sample Output:

```
{
    "user\_name": "jay",
    "created\_at": "2020-02-29 19:30:00"
}
```

`/subscription`

- This is the primary API being tested in this challenge.
- This will need to provide mechanisms to:
    - Register a new subscription for an existing user, with a specified plan and start date

`POST /subscription/`

Inputs:

```
{
    "user\_name": < username >,
    "plan\_id": < plan_id >,
    "start\_date": < date in YYYY-MM-DD format >
}
```

Sample Input

```
{ "user_name": "jay", "plan_id": "PRO_1M", "start_date": "2020-03-03" }
```

Expected Output:

```
{ "status": <["SUCCESS"|"FAILIURE"]>, "amount": <+/- amount
credited/debited> }
```

Sample Output:

```
{ "status": "SUCCESS", "amount": -200.0 }
```

Additional Details

- On success, return 200 HTTP status. For failures, pick an appropriate HTTP code.
- The timestamp indicates the start date for the new plan, and it will be valid for the number of days shown in the table below
- plan_id can be one of those listed in the table below:

Plan ID	Validity (in days)	Cost (USD)
FREE	Infinite	0.0
TRIAL	7	0.0
LITE_1M	30	100.0
PRO_1M	30	200.0
LITE_6M	180	500.0
PRO_6M	180	900.0

`GET /subscription/< username >/< date >`

Sample Input:

`/subscription/jay/2020-02-29`

Note that the date in the above is optional:

`/subscription/jay`

Expected Output:

- **When input date is specified**
  - plan_id that will be active for user at specified date.
  - Number of days left in plan from the specified input date
  - **Sample output**
    `{ "plan_id": "PRO_1M", "days_left": 3 }`
- **When input date is NOT specified**
  - List all subscription entries available in database for user with start and valid till dates.

- **Sample output**


```
[
    {
        "plan_id": "TRIAL",
        "start_date": "2020-02-22",
        "valid_till": "2020-02-28"
    },
    {
        "plan_id": "PRO_1M",
        "start_date": "2020-02-29",
        "valid_till": "2020-03-30"
    }
]
```