Due Date: May 16th , 2020

Setup

- Create a Github repository. Check-in your code
- Write a README.md with all the instructions to install, test and run your code.
- Deploy your code to Heroku/Digital Ocean/AWS/service of your choice and send us your app URL.

Relaxation

- You can use any coding language
- Use any SQL or NoSQL for primary storage or redis for caching layer

Simple API Service

Build a simple API service that exposes the following 2 APIs that accepts JSON data as input. Use only 1 appropriate HTTP method (GET/POST/PUT/DELETE) for your APIs. If the API request is sent using any other HTTP method apart from the one you chose, return HTTP 405.

Authentication
For all API requests, Basic Authentication is required.
If authentication is failing, return proper HTTP status code.

API /inbound/sms/

Input Parameters

Parameter	**required**	**example**
from (min length 6, max length 16)	true	1234567890
to (min length 6, max length 16)	true	12345
text (min length 1, max length 120)	true	Hello from Telstra, STOP, ...

Expected API behavior

- Input parameters should be valid
- When text is STOP or STOP\n or STOP\r or STOP\r\n
    - The 'from' and 'to' pair must be cached with an expiry of 4 hours.

Output JSON response
If required parameter is missing:
{"message": "", "error": "<parameter_name> is missing"}

If parameter is invalid:
{"message": "", "error": "<parameter_name> is invalid"}

Any unexpected error:
{"message": "", "error": "unknown failure"}

If all parameters are valid:
{“message”: “inbound sms is ok”, “error”: “”}

API /outbound/sms

Input Parameters

Parameter	**required**	**example**
from (min length 6, max length 16)	true	1234567890
to (min length 6, max length 16)	true	12345
text (min length 1, max length 120)	true	Hello from telstra, Hello World

Expected API behavior

- Input parameters should be valid
- If the pair of ‘from’ and ‘to’ matches the cached pair(STOP), return an error response with appropriate HTTP status code (see Output JSON response below)
- Do not allow more than 50 API requests using the same ‘from’ number in 1 hour. Return an error response with appropriate HTTP status code in case the limit has been reached (see Output JSON response below)

Output JSON response
If required parameter is missing:
{“message”: “”, “error”: “<parameter_name> is missing”}

If parameter is invalid:
{“message”: “”, “error”: “<parameter_name> is invalid”}

If the pair ‘to’ and ‘from’ matches the cached pair:
{“message”: “”, “error”: “sms from <from> and to <to> blocked by STOP request”}

If 50 requests limit reached in last 1 hour with same ‘from’ parameter:
{“message”: “”, “error”: “limit reached for from <from>”}

Any unexpected error:
{“message”: “”, “error”: “unknown failure”}

If all parameters are valid:
{“message”: “outbound sms is ok”, “error”: “”}

Tests

- Write unit tests for all the functions in your code

Tips:

1. API should do input validations
2. Emphasize on coding principles (code modularity, separation of concerns, testability)

3. Appropriate unit tests
4. Return proper HTTP status codes in case of error
5. Handle all the edge/error conditions