Testing

Strategy

For the backend, we will have another instance of software running in a isolated docker container isolated with our production environment for unit testing. The unit test runs with pytest, a Python test framework, and it can run automatically. Since the backend essentially is providing RESTful API to the client, the way we test the API is by send HTTP request to our testing server to verify responses' status code and content. By doing so, we can easily implement new feature, refactor code, or make any other changes, then test it with existing test cases.

Test Cases (Backend)

Usage

The test cases is located under the `tests` folder.
`$ make test`

Sample Console Output

```
============================================= test session starts ==============================
platform linux -- Python 3.5.2, pytest-3.0.6, py-1.4.32, pluggy-0.4.0
rootdir: /home/app, inifile:
plugins: ordering-0.4
collected 6 items

test/test_list.py .
test/test_article.py .
test/test_list.py ....

========================================== 6 passed in 0.32 seconds ============================
```

Test Case 1

Item	Detail
Name	`test_create_user_list`
Purpose	Verify that user would be able to create a reading list.
Reference	US 2.1

Test Case 2

Item	Detail

Purpose	Verify that user would be able to get his reading lists.
Reference	US 2.1

Test Case 3

Item	Detail
Purpose	Verify that user would be able to create an article under a personal list.
Reference	US 2.5 US 2.3 US 3.1

Test Case 4

Item	Detail
Purpose	Verify that user would be able to delete an article under a personal list.
Reference	US 3.8

Test Case 5

Item	Detail
Purpose	Verify that user would be able to archive a personal list.
Reference	US 3.9

Test Case 6

Item	Detail
Purpose	Verify that user would be able to retrieve a archived personal list.
Reference	US 3.10

Test Case 7

Item	Detail
Purpose	Verify that user can post comment to a article
Reference	US 3.2

Test Case 8

Item	Detail
Purpose	Verify that user can create a reading group
Reference	US 2.2