Textbook Town API Guide

Prepared March 1, 2017

Introduction:

This document will outline the API interface between the front-end and back-end systems of Textbook town. Functional frontend expectancies are located in /webroot/api for frontend testing purposes.

Overview of Call Structure:

- SERVER: refers to the server on which the backend server is running. It may be localhost:8080 if on a local testing machine. On production it will be DOMAIN_NAME:8080 while the frontend will be served on DOMAIN_NAME (with port :80 implicit)
- All API endpoints will be located at SERVER/api/

User/Register:

DESC: Allows new users to register for the system.

URL: SERVER/user/register

METHOD: POST

HEADERS: {

Content-Type: application/json

}

BODY: {

username: ‘nameHere’,

password1: ‘passwordHere’,

password2: ‘matchingpassword’

contact: ‘nelder@uwo.ca’

}

Note that contact could be email or facebook messenger link, etc.

SUCCESS RESPONSE: {

status: ‘success’

}

FAILURE RESPONSE: {

status: ‘failure’,

message: ‘username_taken’ or ‘username_too_short’ or ‘username_too_long’ or ‘password_too_short’ or ‘passwords_not_matching’ or ‘missing_arguments’

}

Where message can be: **username_taken** for username already exists, **username_too_long** for username > 32 chars, **password_too_short** or **passwords_not_matching** for password is too short or does not match.

User/Login:

DESC: Allows new users to login to the system.

URL: http://username:password@SERVER/user/login

METHOD: GET

HEADERS: {

Content-Type: application/json

}

(there might be an implicit authorization header created when you put the username and password in the URL like that)

SUCCESS RESPONSE: {

status: 'success',

token: 'token_for_login_to_be_stored'

duration: 'token_duration'

}

FAILURE RESPONSE:

(for invalid username/password)

HEADERS: {

```
Content-Length →19
Content-Type →text/html; charset=utf-8
Date →Thu, 02 Mar 2017 01:55:01 GMT
Server →Werkzeug/0.9.4 Python/3.4.3
WWW-Authenticate →Basic realm="Authentication Required"
```

}

HTTP status is 401 unauthorized

BODY: {

'Unauthorized Access'

}

isAuth:

DESC: Checks whether or not a token is valid

URL: http://SERVER/login/check

METHOD: POST

HEADERS: {

Content-Type: application/json

}

BODY: {

token: ‘auth_token’,

}

SUCCESS RESPONSE: {

status: ‘success’

}

FAILURE RESPONSE: {

status: ‘failure’

}

Book/Search:

DESC: Fetches books in system by parameters or defaults to recent first if none specified.

URL: SERVER/book/search

METHOD: GET

ARGUMENTS: ?q=QUERY_STRING

Note that the query string is a plain text search string of either textbook names or class names.

If there is no query specified, return a default list based on closing date; most recent first.

SUCCESS RESPONSE:

```
{
  "status": "success",
  "books": [
                    {
                          "id":0,
                              "title": "Guided Missile Fundamentals: Actually, it is rocket Science!",
                              "author": "Margret MacMiller",
                              "date_closing": "Feb 22, 2017",
                              "subject": "CS4444",
                              "image": "http://www.site.com/img/dawkjdhjakwhdjkhk12hj3k12h3k1.png",
                              "tag": "",
                              "bids": 15,
                              "price": 129
                    },
                    {
                    "id":1,
                              "title": "Integrative Wildlife Nutrition: A comprehensive guide.",
                              "author": "Jausn Simpson",
                              "date_closing": "Feb 25, 2017",
                              "subject": "CS4444",
                              "image": "http://www.site.com/img/213kh12kj312k3h1khj3jk12hkj31.png",
                              "tag": "Last Day",
                              "bids": 44,
                              "price": 242
                    }
  ]
}
```

Note that images need to be the full URL used to fetch the resource. Note the id should be the DB id such that the id is globally unique.

FAILURE RESPONSE: {

```
        status: 'failure'
}
```

Book/Add:

DESC: Adds a textbook to a system.

URL: http://token@SERVER/book/add

METHOD: POST

HEADERS: {

```
        Content-Type: application/json
}
```

(THIS IS ALL ONE HUGE MULTIPART FORM WITH 4 FILES)

BODY:

```
{
    "title": "Guided Missile Fundamentals: Actually, it is rocket Science!",
    "author": "Margret MacMiller",
    "version": "V12",
    "desc": "text",
    "publisher": "text",
    “year": 2017,
    "isbn": "12231-41-24214124-212312-12",

    "rating": "77",

    "sellby": "yyyy-mm-dd",
    "subject": "CS4444",
    "price": 100,

    "coverpic": "Multipart file",
    "pic1": "Multipart file",
    "pic2": "Multipart file",
    "pic3": "Multipart file"
}
```

SUCCESS RESPONSE: {

```
            status: ‘success’,
            id: 12
}
```

Specifies the book ID as it has been now stored in the DB.

FAILURE RESPONSE (for not logged in user with invalid token):

HEADERS: {

```
Content-Length →19
Content-Type →text/html; charset=utf-8
Date →Thu, 02 Mar 2017 01:55:01 GMT
Server →Werkzeug/0.9.4 Python/3.4.3
WWW-Authenticate →Basic realm="Authentication Required"
```

}

HTTP status is 401 unauthorized

BODY: {

‘Unauthorized Access’

}

FAILURE RESPONSE (if user has valid token but issues with the form data): {

status: 'failure'

message: 'appropriate response to be echoed to user'

}

TEXTBOOK VIEWING / BIDDING ENDPOINTS:

Book/Bid:

DESC: Allows users to place a bid

URL: token@SERVER/book/bid

METHOD: POST

HEADERS: {

}

POST JSON: {

"bid" : 100

"textbook" : textbookID

}

SUCCESS RESPONSE: {

status: 'success'

}

FAILURE RESPONSE (if user has valid token but issues with bid amount): {

status: 'failure'

message: 'bid must be a positive integer' or 'bid too low'

}

Book/HasBid:

DESC: Check if current logged in user has already bid on textbook

URL: token@SERVER/book/hasbid?id=textbookID

METHOD: GET

HEADERS: {

}

SUCCESS RESPONSE: {

status: ‘success’

“hasBid” : True/False

}

FAILURE RESPONSE (if user has valid token but issues with bid amount): {

status: ‘failure’

}

FAILURE RESPONSE (for not logged in user with invalid token):

HEADERS: {

```
Content-Length →19
Content-Type →text/html; charset=utf-8
Date →Thu, 02 Mar 2017 01:55:01 GMT
Server →Werkzeug/0.9.4 Python/3.4.3
WWW-Authenticate →Basic realm="Authentication Required"
```

}

HTTP status is 401 unauthorized

BODY: {

‘Unauthorized Access’

}

Book/BuyerCheck

DESC: Check whether current user is a buyer or seller

URL: http://token@SERVER/book/buyercheck?id=textbookID

METHOD: GET

HEADERS: {

[Include whatever the usual token passing header is]

}

SUCCESS RESPONSE: {

status: ‘success’

“isBuyer” : true/false

}

- If true should make a call to the buyer view of textbook endpoint
- If false should make a call to the seller view of textbook endpoint

FAILURE RESPONSE (for not logged in user with invalid token):

HEADERS: {

```
Content-Length →19
Content-Type →text/html; charset=utf-8
Date →Thu, 02 Mar 2017 01:55:01 GMT
Server →Werkzeug/0.9.4 Python/3.4.3
WWW-Authenticate →Basic realm="Authentication Required"
```

}

HTTP status is 401 unauthorized

BODY: {

‘Unauthorized Access’

}

Book/Info

DESC: Data grab for the buyer view of textbook page

URL: SERVER/book/info?id=textbookID

METHOD: GET

HEADERS: {

}

SUCCESS RESPONSE: {

```
"status": "success",
"isCurrent": true,
"auction": 8,
"author": "Mac",
"averagePhoto": "http://127.0.0.1:5000/img/03da7414-1110-4acf-a25b-8ff1eba61de9.jpg",
"bestPhoto": "http://127.0.0.1:5000/img/4e6a8a08-33ca-4b4a-ac07-c8ea1eb2fdd1.png",
"closingDate": "Mar 20, 2017",
"condition": 60,
"course": "CS 2212",
"coverPhoto": "http://127.0.0.1:5000/img/f6ec7f90-04d5-4777-af98-023c0a160a3e.png",
"description": "cool book",
"id": 8,
"isbn": "xxx-xxx",
"minimumBid": 55,
"publisher": "random house",
            "seller": 4,
            "title": "This is my awesome book about engineering",
            "version": "1",
            "worstPhoto": "http://127.0.0.1:5000/img/7fd104cf-5ef9-41d0-9a59-651afb1075f3.jpg",
            "yearPublished": 2016

}
```

FAILURE RESPONSE (for not logged in user with invalid token):

-- maybe redirect to the login page, and then back here if this is the case?

HEADERS: {

```
Content-Length →19
Content-Type →text/html; charset=utf-8
Date →Thu, 02 Mar 2017 01:55:01 GMT
Server →Werkzeug/0.9.4 Python/3.4.3
WWW-Authenticate →Basic realm="Authentication Required"
```

}

HTTP status is 401 unauthorized

BODY: {

'Unauthorized Access'

}

Book/SellerInfo

DESC: Data grab for the seller view of textbook page

URL: SERVER/book/sellerInfo?id=textbookID

METHOD: GET

HEADERS: {

Include the standard auth header with token to verify the user

}

SUCCESS RESPONSE: {

```
            "status": "success",
            "isCurrent": false
```

(If isCurrent is true, this bids list won't be included, but you won't have to worry about that")

```
           "bids": [
              {
```

```
        "bid": 135,
        "profile_link": "hategrails@hategrails.com",
        "user_name": "hategrails"
      },
      {
        "bid": 130,
        "profile_link": "call me",
        "user_name": "pierce"
      },
      {
        "bid": 115,
        "profile_link": "facebook",
        "user_name": "abdulla"
      }
     ],

    "auction": 8,
    "author": "Mac",
    "averagePhoto": "http://127.0.0.1:5000/img/03da7414-1110-4acf-a25b-8ff1eba61de9.jpg",
    "bestPhoto": "http://127.0.0.1:5000/img/4e6a8a08-33ca-4b4a-ac07-c8ea1eb2fdd1.png",
    "closingDate": "Mar 20, 2017",
    "condition": 60,
    "course": "CS 2212",
    "coverPhoto": "http://127.0.0.1:5000/img/f6ec7f90-04d5-4777-af98-023c0a160a3e.png",
    "description": "cool book",
    "id": 8,
    "isbn": "xxx-xxx",
    "minimumBid": 55,
    "publisher": "random house",
    "seller": 4,
    "title": "This is my awesome book about engineering",
    "version": "1",
    "worstPhoto": "http://127.0.0.1:5000/img/7fd104cf-5ef9-41d0-9a59-651afb1075f3.jpg",
    "yearPublished": 2016

}
```

FAILURE RESPONSE (if user does not own textbook--this shouldn't be an issue if people are using the site properly, but it's an extra safeguard):

```
{
  "message": "you are not the seller of this book”,
  "status": "failure"
}
```

FAILURE RESPONSE (for user with invalid login token):

HEADERS: {

```
Content-Length →19
Content-Type →text/html; charset=utf-8
Date →Thu, 02 Mar 2017 01:55:01 GMT
Server →Werkzeug/0.9.4 Python/3.4.3
WWW-Authenticate →Basic realm="Authentication Required"
```

}

HTTP status is 401 unauthorized

BODY: {

‘Unauthorized Access’

}

For displaying buyer/seller view of textbook:

(Remember that users can search without logging in, but will need to login in to view a book since we need to know if they are buyer or seller to render appropriate page)

- **This is what I was envisioning for calls to backend:**
-

1. **Make a super simple call to find out if the current user is a buyer or seller of the given textbook**
2. **Based on this information, make the appropriate call to the “BuyerView” endpoint or “SellerView” endpoint**
    a. **In “BuyerView” call, backend will send back whether bidding is open or closed AND whether or not the current user has already placed a bid on the textbook, as well as all textbook data that needs to be displayed**
    b. **In “SellerView” call, backend will send back whether bidding is open or closed, as well as all textbook data that needs to be displayed. If “isCurrent” = false, backend will include a list of the top 3 bidders in the JSON**