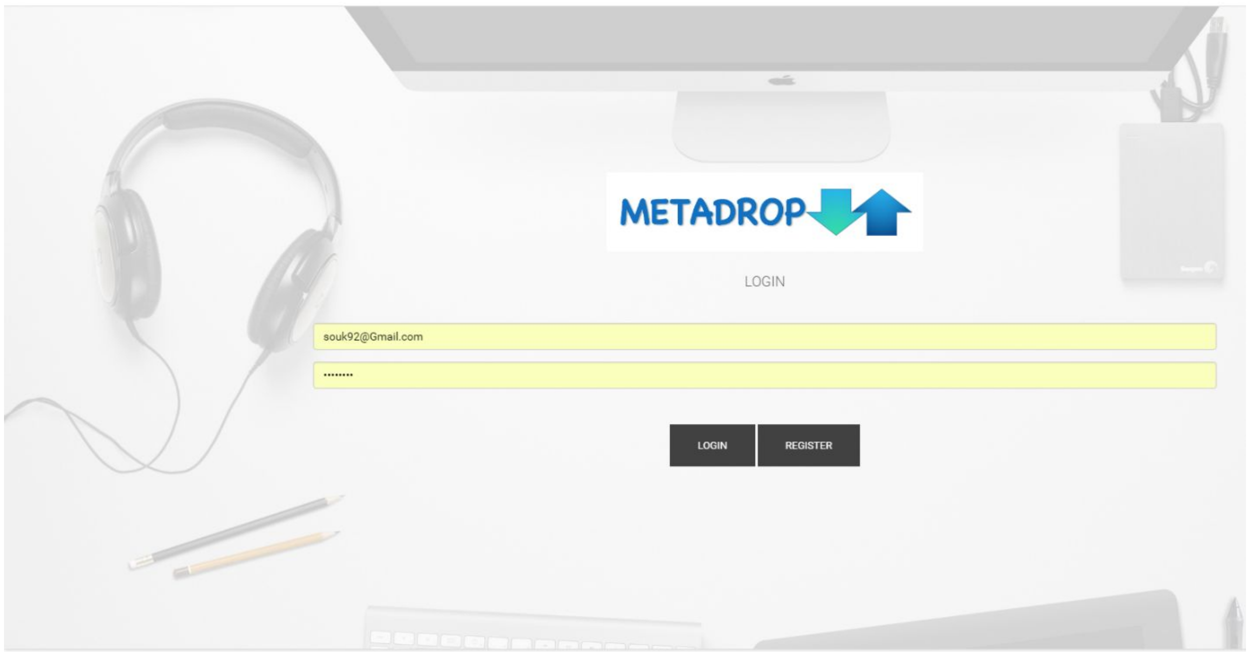
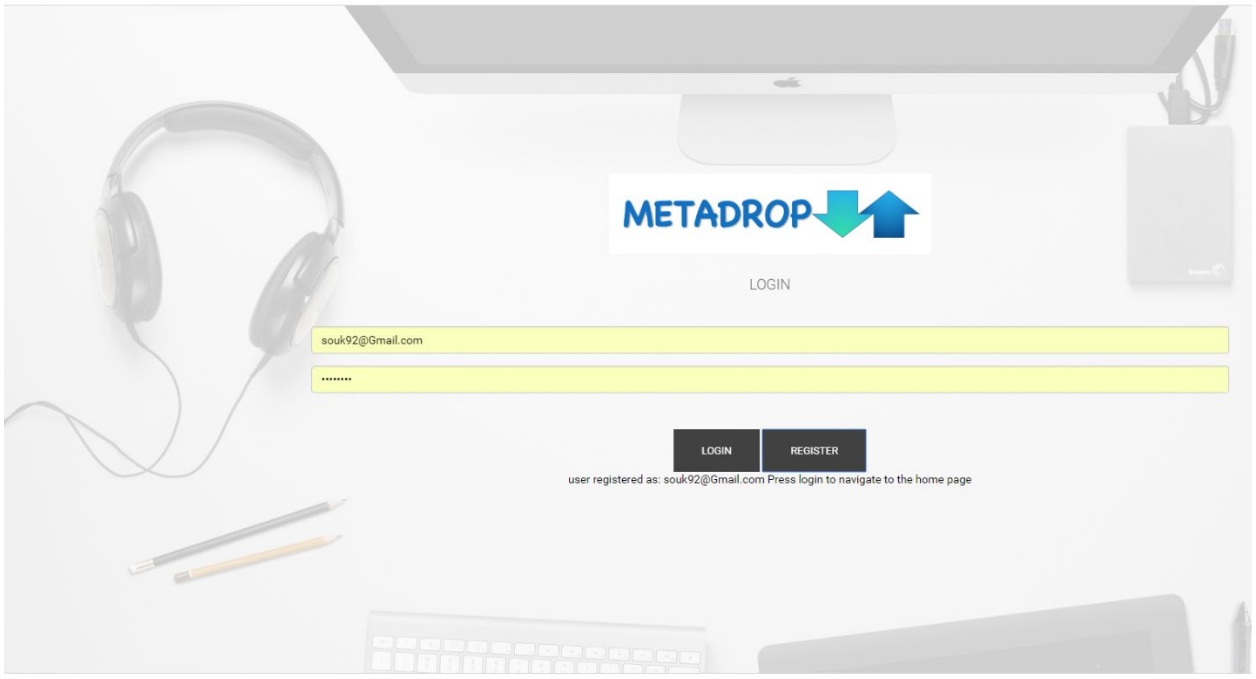
Test Cases



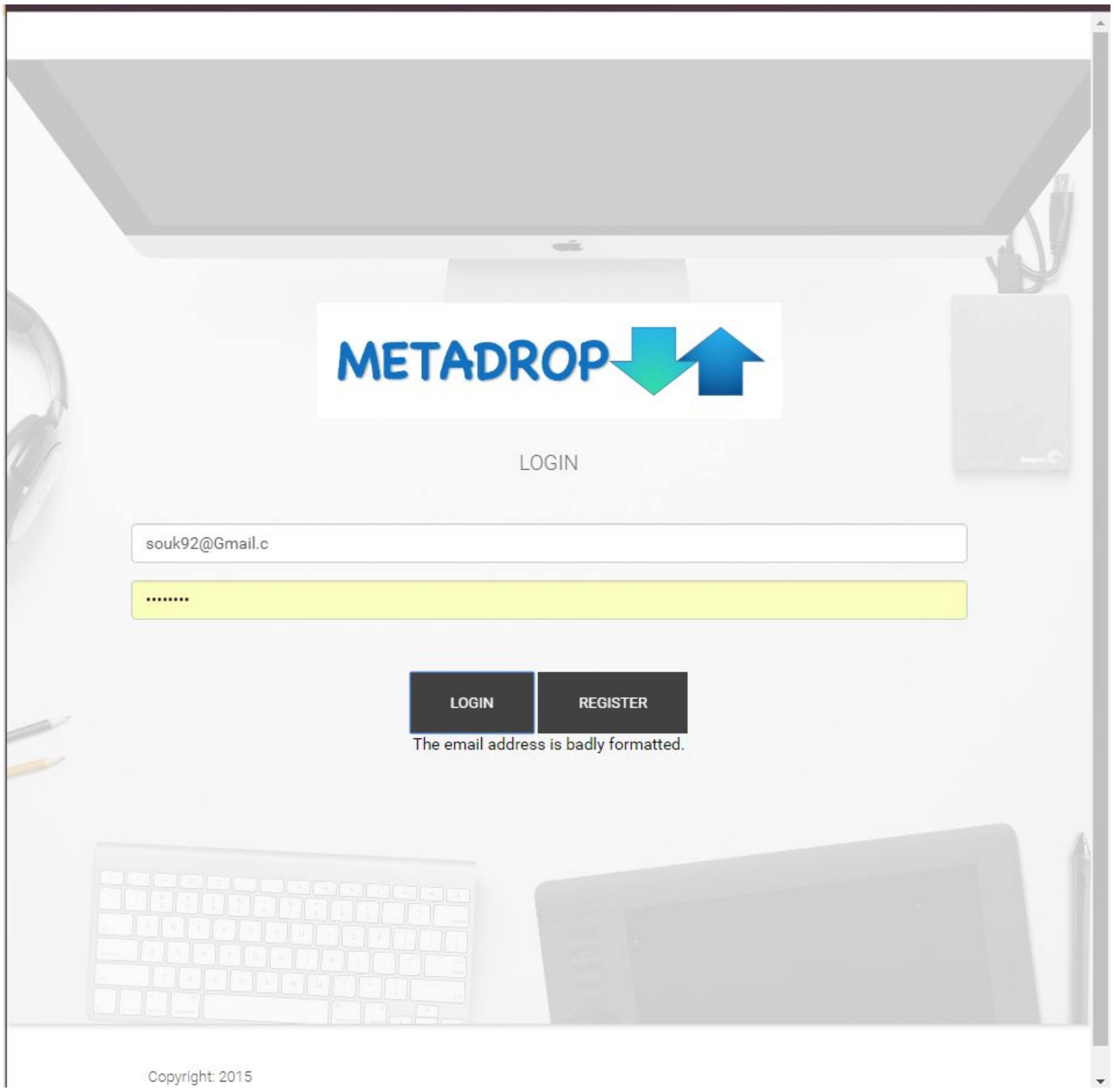


user registered as: souk92@Gmail.com Press login to navigate to the home page

- Above, you'll see a user attempting to register for an account on Metadrop using a username that was previously used. This is part of our **INTEGRATION TESTING (verification).**
    - o Successful case: We expect the user to receive a message saying "user registered as email. Press login to navigate to the home page." – Which means that this

username has already been registered and that the person trying to login must use the right credentials to access our website.

- Failed case: User gets redirected to the home page using a username someone already registered.



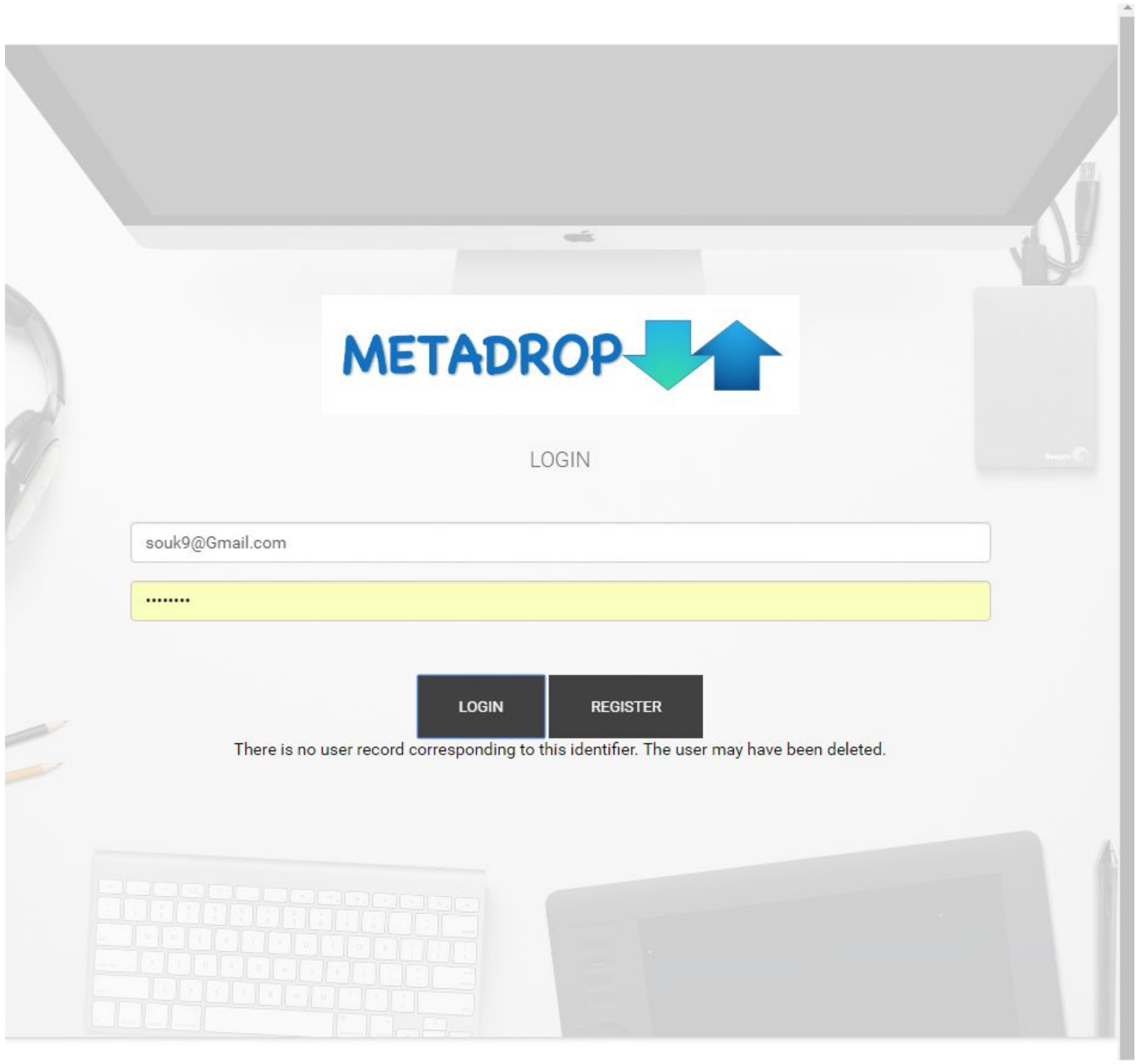
souk92@Gmail.c

••••••••

The email address is badly formatted.

- The picture above shows a user trying to register with an invalid user name. This is another part of our **INTEGRATION TESTING.** What I am referring to here is not putting the right username in where it's required.

- Successful case: We expect to see an error message, “the email address is badly formatted,” our system only accepts usernames in the form of an email address. Therefore it must have a local name, that is separated with an @ symbol, that is separated by a working domain.
- Failed case: The user successfully logs in to Metadrop and redirected to the home page.



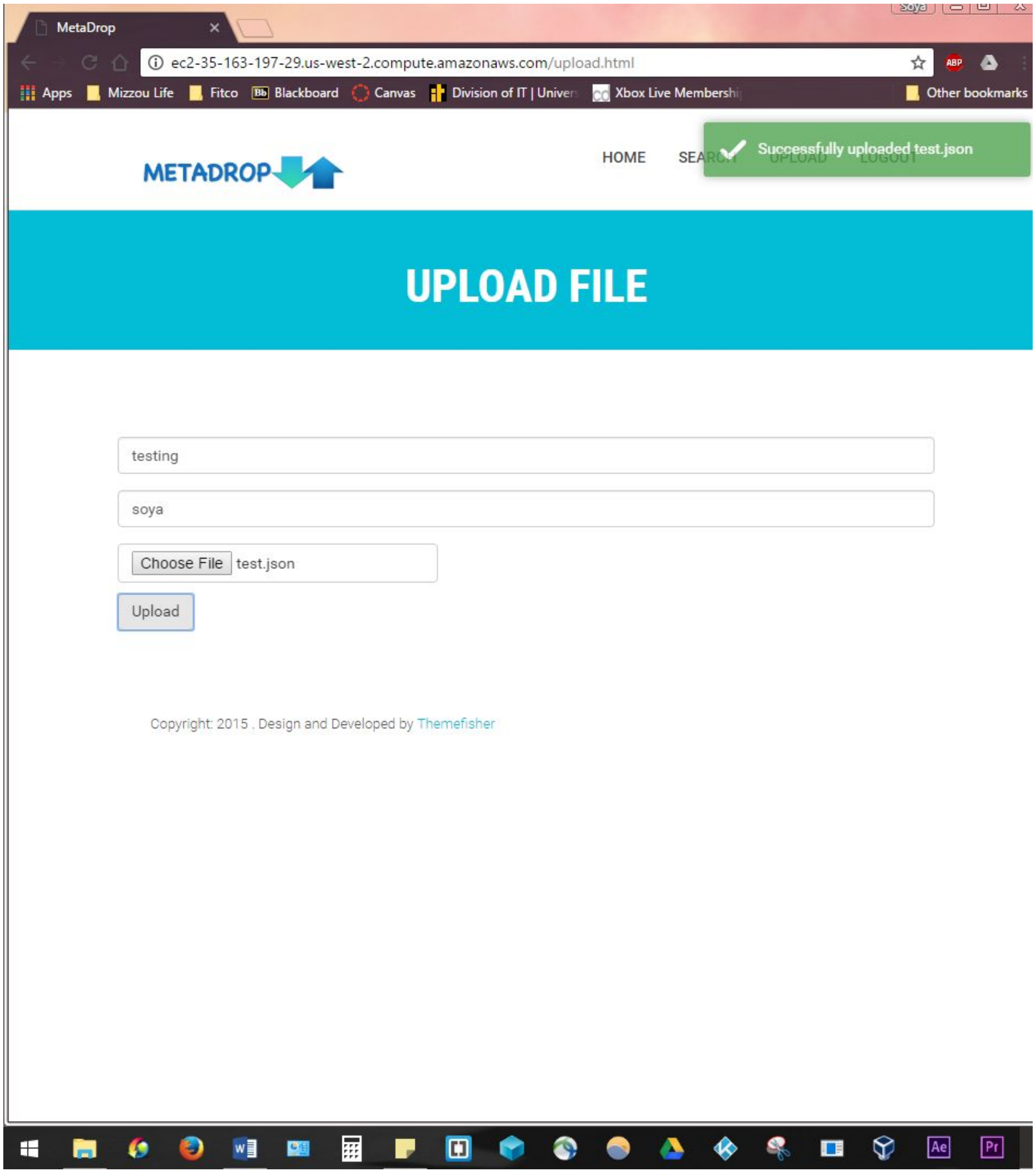
souk9@Gmail.com

••••••••

There is no user record corresponding to this identifier. The user may have been deleted.

- The picture above shows another form of **INTEGRATION TESTING**; here the user attempts to access Metadrop with a username that does not exist.

  - Successful Case: We expect the user to receive a message; “there is no user record corresponding to this identifier. The user may have been deleted.”
  - Failed Case: The user gets access to Metadrop and bypasses our security.



Successfully uploaded test.json

UPLOAD FILE

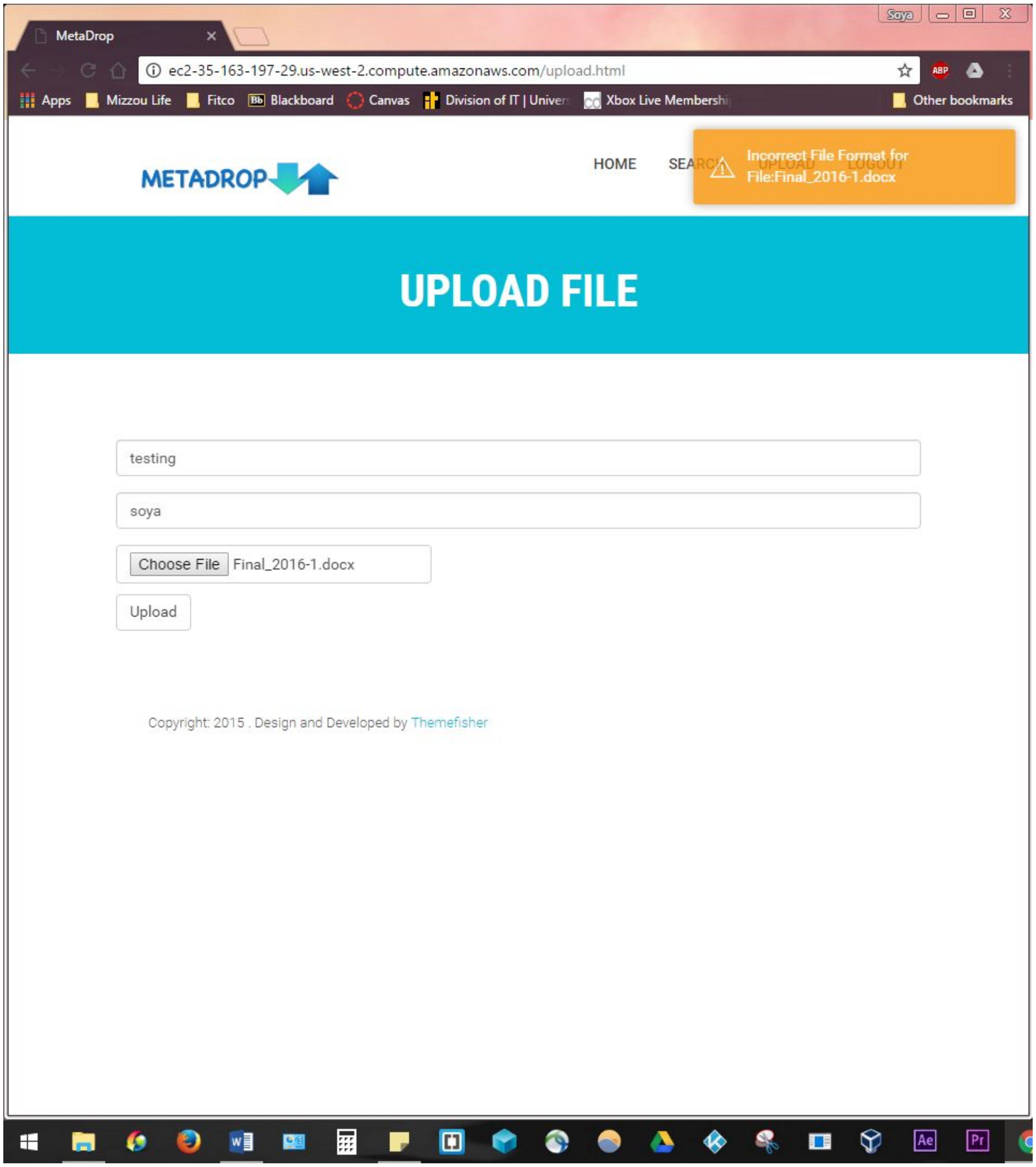
testing

soya

Choose File test.json

Upload

Copyright: 2015 . Design and Developed by Themefisher



Incorrect File Format for File:Final_2016-1.docx

UPLOAD FILE

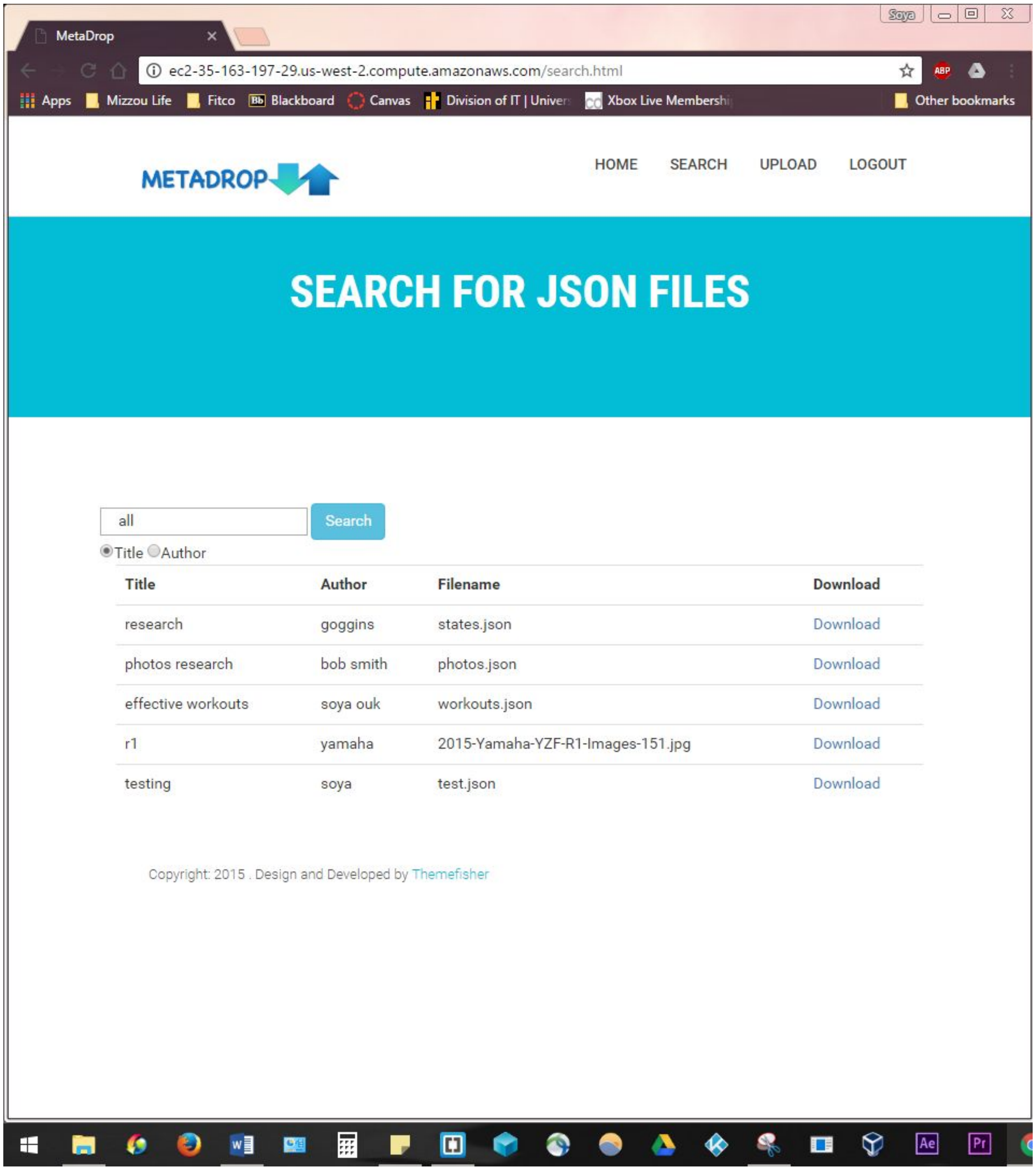
testing

soya

Final_2016-1.docx

Copyright: 2015 . Design and Developed by Themefisher

- Above are pictures of a user uploading a file.
  - Successful Case: We expect the user to be able to upload a file that's of JSON format.
  - Failed Case: User can upload any file type; (docx, pdf, php, html, etc…), but they will get notified saying that it's 'invalid.'



SEARCH FOR JSON FILES

all [Search]

[✓ Title] Author

Title	Author	Filename	Download
research	goggins	states.json	Download
photos research	bob smith	photos.json	Download
effective workouts	soya ouk	workouts.json	Download
r1	yamaha	2015-Yamaha-YZF-R1-Images-151.jpg	Download
testing	soya	test.json	Download

Copyright: 2015 . Design and Developed by Themefisher

The Search Function

Success Case: User is able to view the available files for download and view the json manifest.

Failed Case: User can view the files that are listed but cannot download to view the actual data.

- We will also try to test edge cases such as uploading a file that is over our max limit (250MB); **fail case:** uploading a file that's 1GB which is 4X the limit. And **success case:** if the user gets an error that says, "the file is too large." Unfortunately, we don't have that part of the code figured out yet so we haven't done it yet.
- Another possible edge case would be a **Success:** if the user is only allowed to type 300 words in the text field when uploading a file and writing its' description. It would **fail** (of course) if the user uploads a file with a comment that has 300+ words, (4000 or more). **WE WERE NOT** able to implement this portion of the code since we were pressed for time, so we didn't really test for edge cases.

Just about every test described above are mainly Integration Testing.

User Acceptance Testing (validation) will be implemented by Dr. Goggins and possibly the TA's. They will be given all the documents including the; README, Requirements Analysis, User Manual, etc. They will then go through the entire Validation process making sure that what they asked for is what's promised by us.

Unit Testing (verification) is a testing method used to assess individual units or parts of the source code to see if they meet the requirements and are fit for use. When referring to Metadrop, we should be able to test any function or class of the source code individually to determine this. We're verifying that all the units/functions work as expected. Unit testing should be implemented every time you update your code.

We were able to perform a bunch of unit tests after the fourth sprint. We made sure to test the functionality of certain buttons on the upload page:

- once the user is on the upload page, they'll see add a title, author text box and a "choose file" button.
    - here we made sure that the default file type is json, that way when they're searching for a json file to upload, they won't see a whole bunch of other file types; docx, aep, ppt, etc.

  success case: if the user chooses a file type that is anything other than json, toastr will pop up an orange notification letting the user know that it's an "incorrect file type." If the user does find a json file to upload, they should get a green notification if the upload succeeds.

  failure case: you don't get a warning for picking an incorrect file type, it successfully accepts jpeg, gifs, docx, etc. files as a json.

We were able to perform unit testing on our search page as well:

- our system loads all of the files, then if the user wants to narrow down the search, all they have to do is type in either the title or author name.
    - an example: title: r1, author: yamaha. if the user searches r1 and forgets to check the title radio, our function won't locate anything.

      success case: user searches specific file names by "title" or "author" and gets results.

      failure case: the user can search any key term and it'll pop up regardless of which radio button is checked: title, or author.

We also tested basic functionality:

- make sure the home button takes you to http://ec2-35-163-197-29.us-west-2.compute.amazonaws.com/home.html
- make sure the search button takes you to http://ec2-35-163-197-29.us-west-2.compute.amazonaws.com/search.html
- make sure the upload button takes you to http://ec2-35-163-197-29.us-west-2.compute.amazonaws.com/upload.html
- make sure the logout button logs you out.

  success: user logs out and is redirected to the login page http://ec2-35-163-197-29.us-west-2.compute.amazonaws.com/index.html

  failed case: user logs out but can hit the back button and still access any of the 3 pages by simply typing in the desired destination address in the search bar of the web browser.