# Manual Testing Procedure

## Table Of Contents

# Account Creation and Log in

## Testing Account Creation

To create an account the user clicks ‘Get Started’ from the nav bar, or clicks ‘Get Started’ on the button contained within the Jumbotron on the main page. Reference Figure 1 and Figure 2.

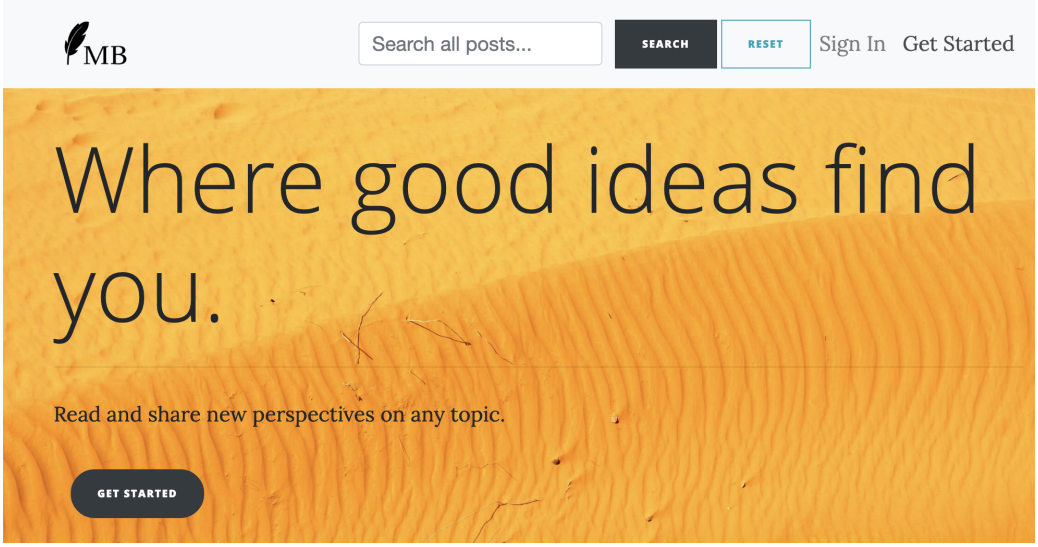


Figure 1: Click ‘Get Started’ to create an account.

Figure 2: Sign up page presented to the user

1. Clicking on ‘REGISTER’ with none of the fields filled in presents the following feedback to the user. Figure 3.

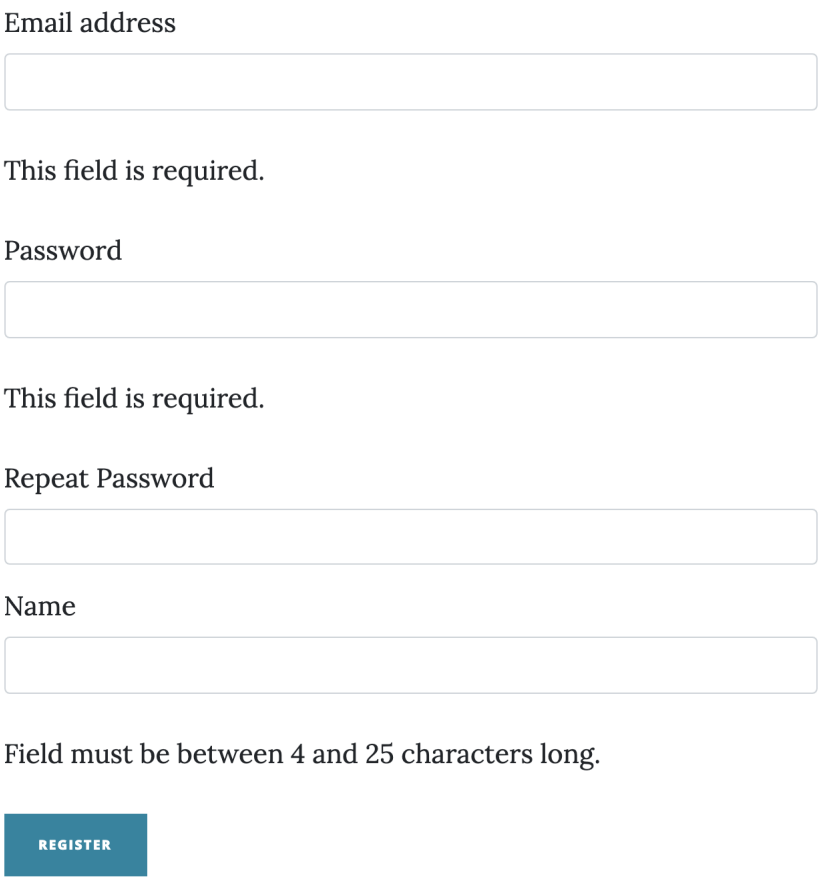


Figure 3: Validation feedback on submission of a blank form.

2. Filling in the email address field with an ‘incorrect’ email type presents the following feedback. Figure 4.


Figure 4: ‘Email address’ must correspond to a valid email regular expression.

3. Filling in a ‘Password’ and a ‘Repeat Password’ that do not match presents the following feedback. Figure 5.

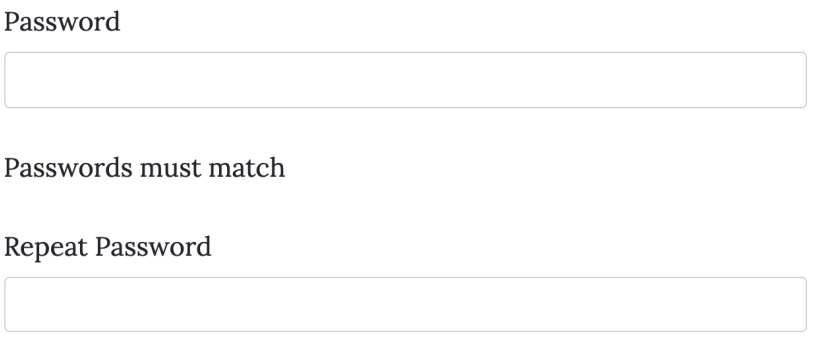


Figure 5: ‘Passwords must match’ feedback to the user.

4. Filling in a ‘Name’ that is less than 4 characters (or greater than 25) returns the following feedback. Figure 6.

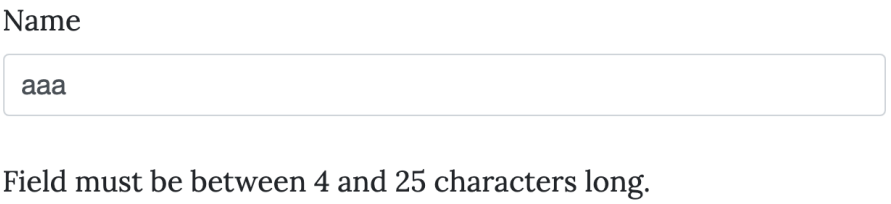


Figure 6: ‘Name’ must be between 4 and 25 characters long.

5. Upon Successful Registration, a profile page is presented. Figure 7.


Registration Successful

## Your Posts

| # | Image | Title | Date |
|---|---|---|---|

Figure 7: Upon successful Registration, a profile page is presented to the new user. Note the ‘Registration Successful’ message that displays.

6. We can confirm that the new user *foobar* has been added to the database by navigating to *'cloud.mongodb.com'*. Figure 8.


```
_id: ObjectId("609388d72fa3292824ddc1ab")
email: "foobar@foo.com"
password: "pbkdf2:sha256:150000$cflaAWqS$3806685db8c591b4a145f1983af5d699e55ff6bb..."
name: "foobar"
```


Figure 8: User *foobar* has been created and added to the database.

## Testing Sign In

We can re-use the same account just created in the previous section to test Sign In. From the home page click on the 'Sign In' navbar link. The following page is presented. Figure 9.

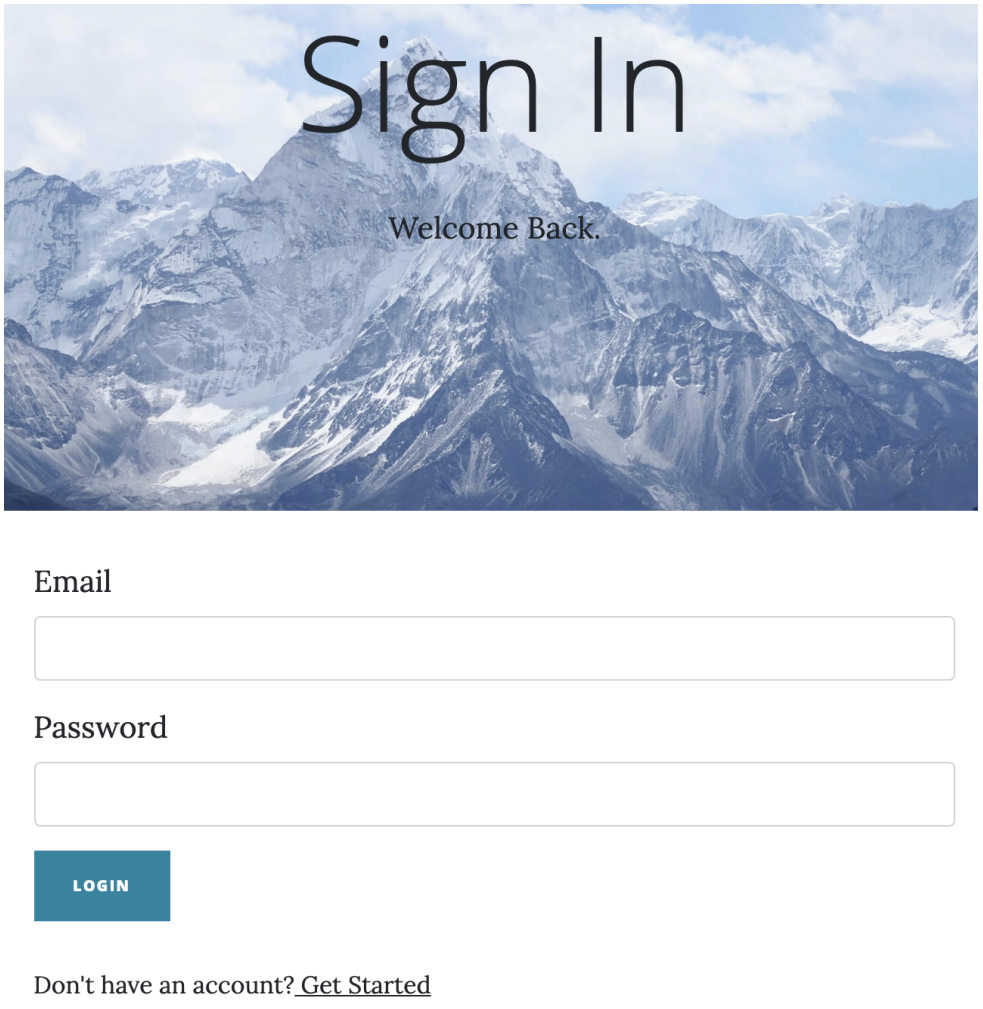


Figure 9: 'Sign In' page presented to the user.

1. Clicking ‘LOGIN’ with none of the fields filled in presents the following feedback to the user. Figure 10.

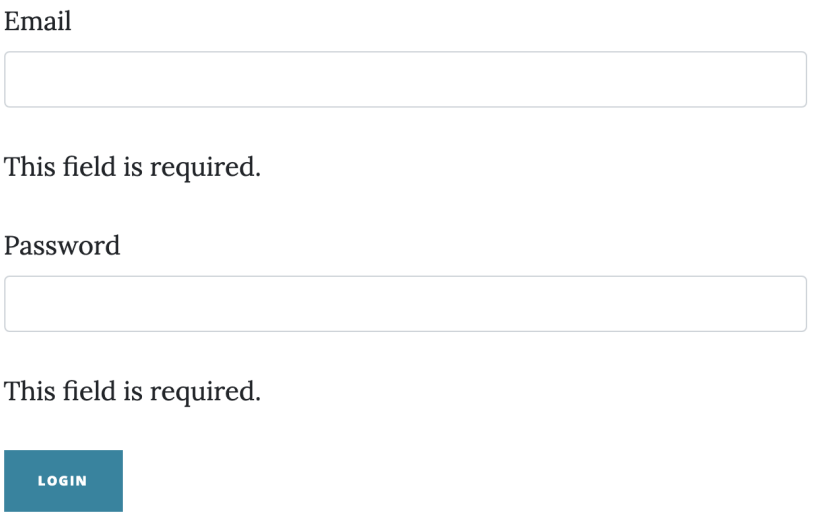


Figure 10: Validation feedback on submission of a blank form.

2. Filling in the ‘Email’ field with an incorrect ‘Password’ presents the following feedback. Figure 11.

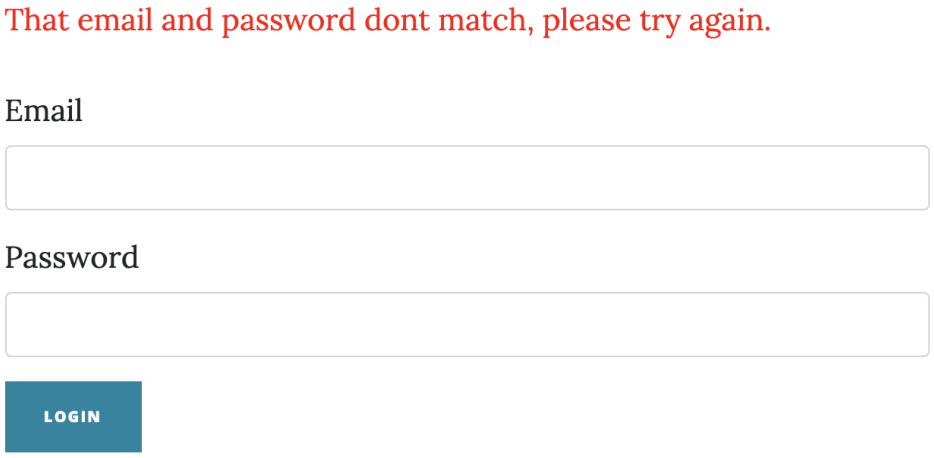


Figure 11: Validation feedback on submission of an incorrect password.

3. Filling in the ‘Email’ field and ‘Password’ field with a non-existent account presents the following feedback. Figure 12.

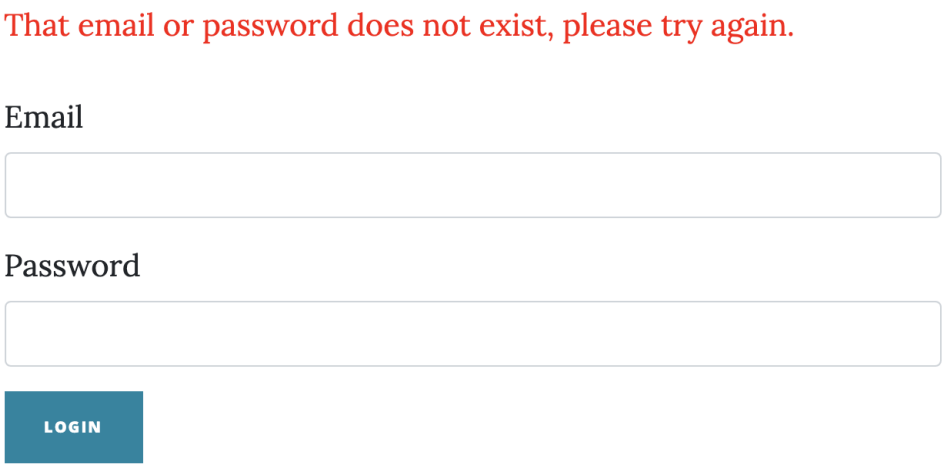


Figure 12: Validation feedback on submission of a non-existent email.

4. A correct combination of ‘Email’ and ‘Password’ brings the user back to the Profile Page shown in Figure 7.

# Testing Create, Read, Update, Delete of Blog Posts

## Create a Post

1. Create a new Post by initially clicking on ‘Write’ within the profile navbar. Figure 13.

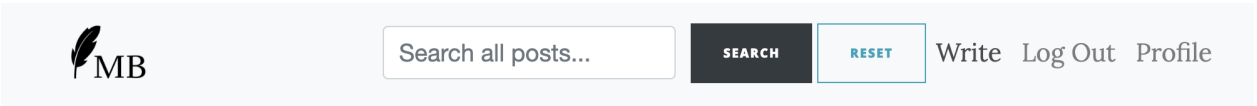


Figure 13: Create a new post by clicking on ‘Write’

2. A ‘Create Post’ page will present itself. Figure 14.

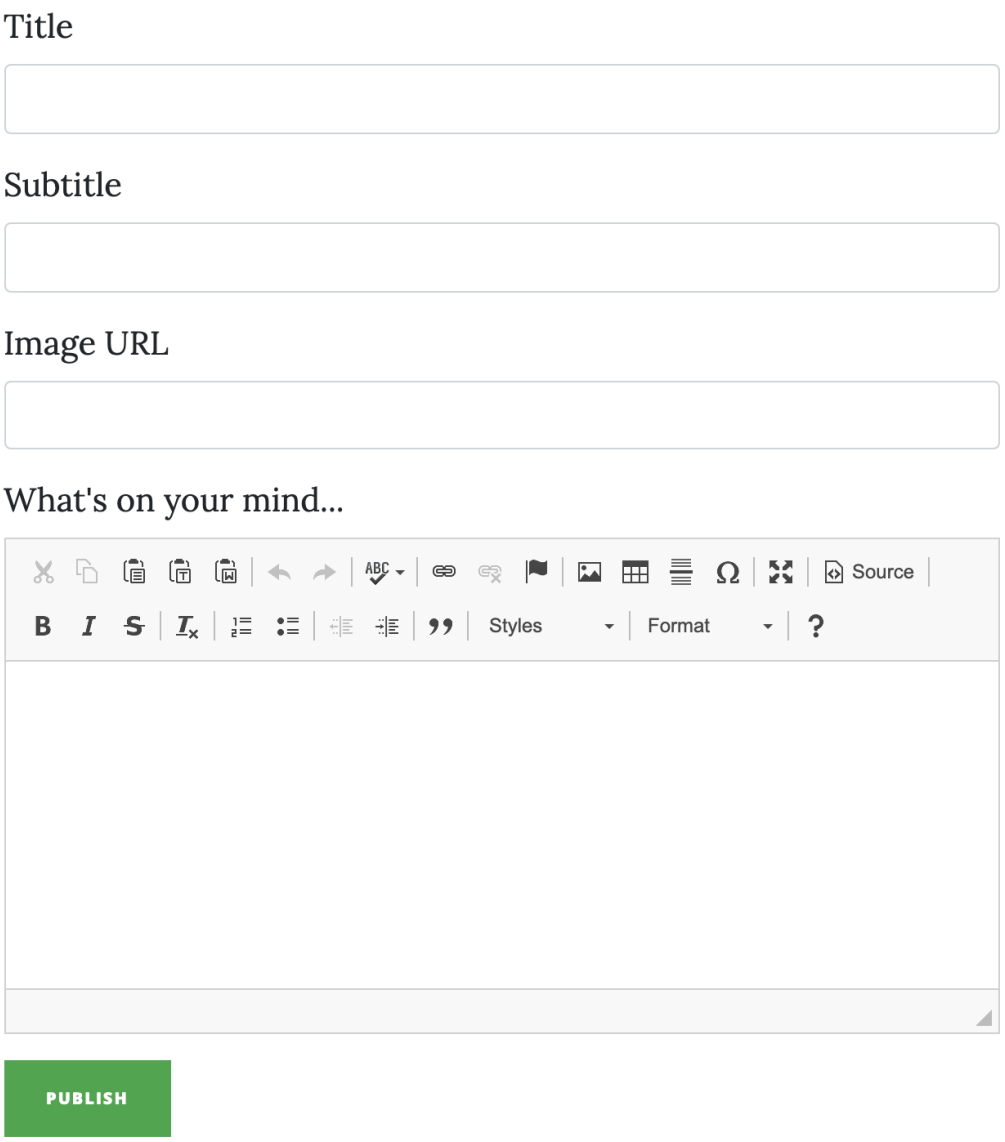


Figure 14: Form for submitting a new Post.

3. Data validation is present on each of the ‘Title’, ‘Subtitle’, ‘Image URL’, ‘What’s on your mind…’ fields, and we can test this by trying to ‘Publish’ a form with none of the fields filled in. Figure 15.

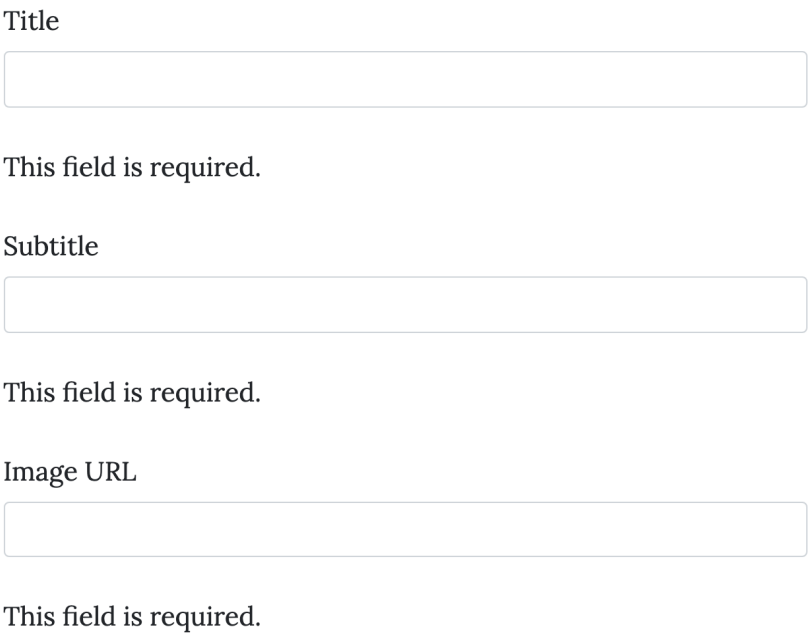


Figure 15: Validation feedback presented to the user on submission of blank form.

4. We can fill out the form with sample data. The ‘Image URL’ field also performs a simple regular expression check to ensure it is a valid URL. Figure 16.


Figure 16: Validation feedback for an ‘Invalid URL’

5. Once all the fields are valid, we can click ‘Publish’ and we are redirected back to the home page. Scrolling down we can see our new post. Figure 17.

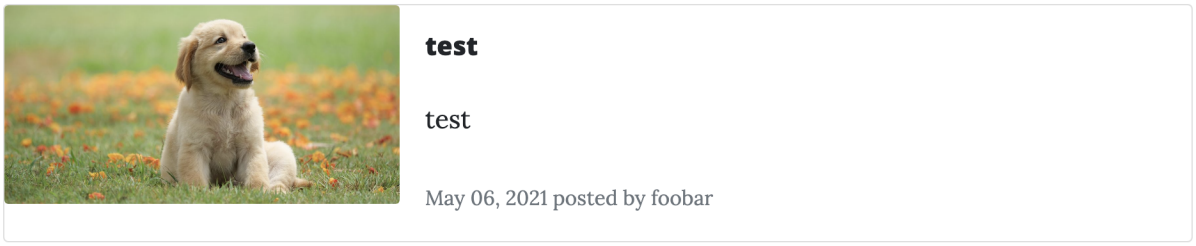


Figure 17: A new Post has been Created using our *‘foobar’* account.

6. We can confirm that the new Post has been added to the database by navigating to *'cloud.mongodb.com'*. Figure 18.


```
_id: ObjectId("609391172fa3292824ddc1ac")
title: "test"
subtitle: "test"
body: "<p>My new post<strong> </strong></p>
"
img_url: "https://hips.hearstapps.com/hmg-prod.s3.amazonaws.com/images/dog-puppy..."
author: "foobar"
date: "May 06, 2021"
```


Figure 18: A new Post has been Created and added to the database.

## Read a Post

1. By clicking on the new post created in the previous section, we are presented with the following. Figure 18.

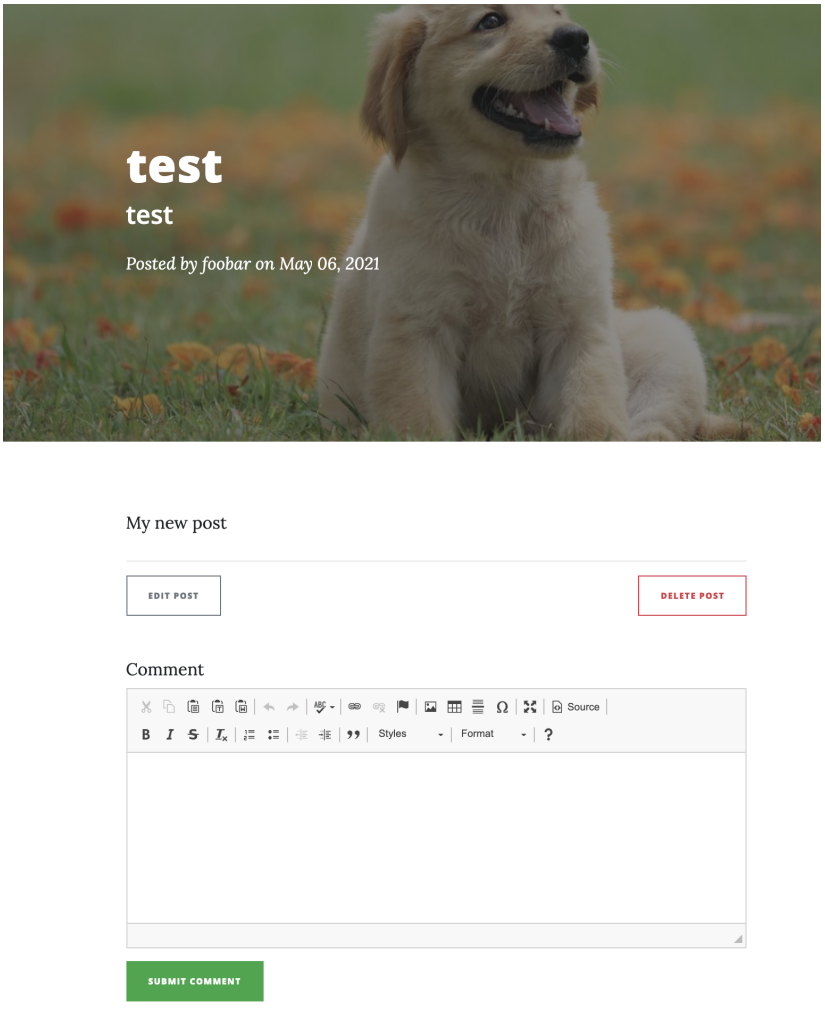


Figure 18: Reading the new Post. As we are logged in, and we are the content creator, we have the options to 'Edit Post' and 'Delete Post'.

2. We were logged in as user ‘foobar’ in step 1. We can test what the page looks like if we are logged out. We click on ‘Log Out’ (which redirects us back to the home page), scroll down to the post, and click on it. We are presented with Figure 19.

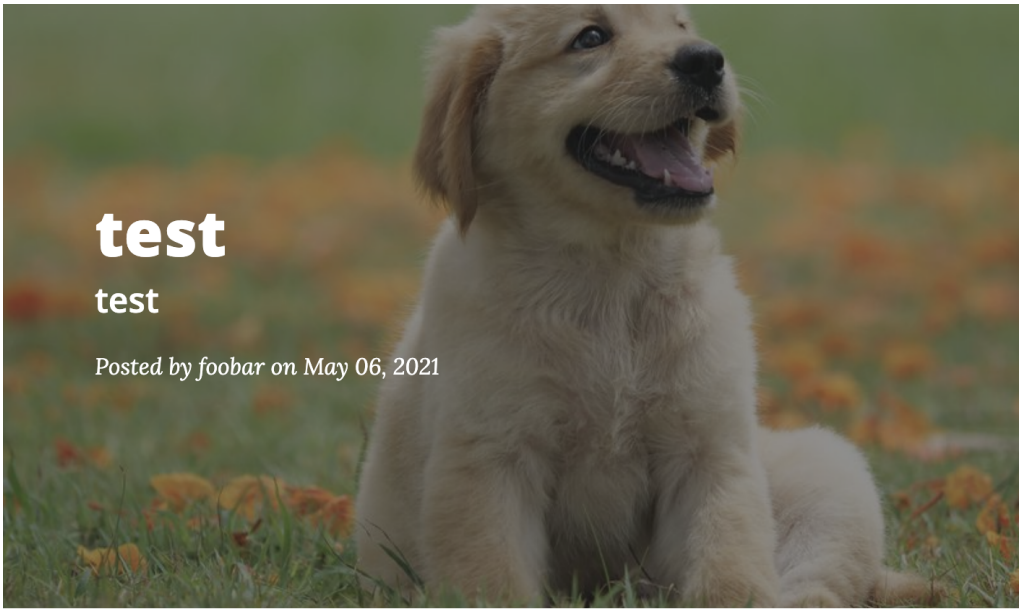


Figure 19: Reading the new Post, as a non-logged in user. The options to ‘Edit Post’ and ‘Delete Post’ are no longer there. The same would also apply if we were logged in as a user that had not created the content.

3. If we were logged in as a completely different user i.e. NOT user ‘*foobar’* we would also see the same as Figure 19. This is because **only the content creator** will have the option to ‘Edit’ or ‘Delete’

## Update a Post

1. We can also view any of our Posts by going to the Profile page. As a logged in user, navigate to ‘Profile’ in the navbar. The user *foobar* is presented with the following. Figure 20.

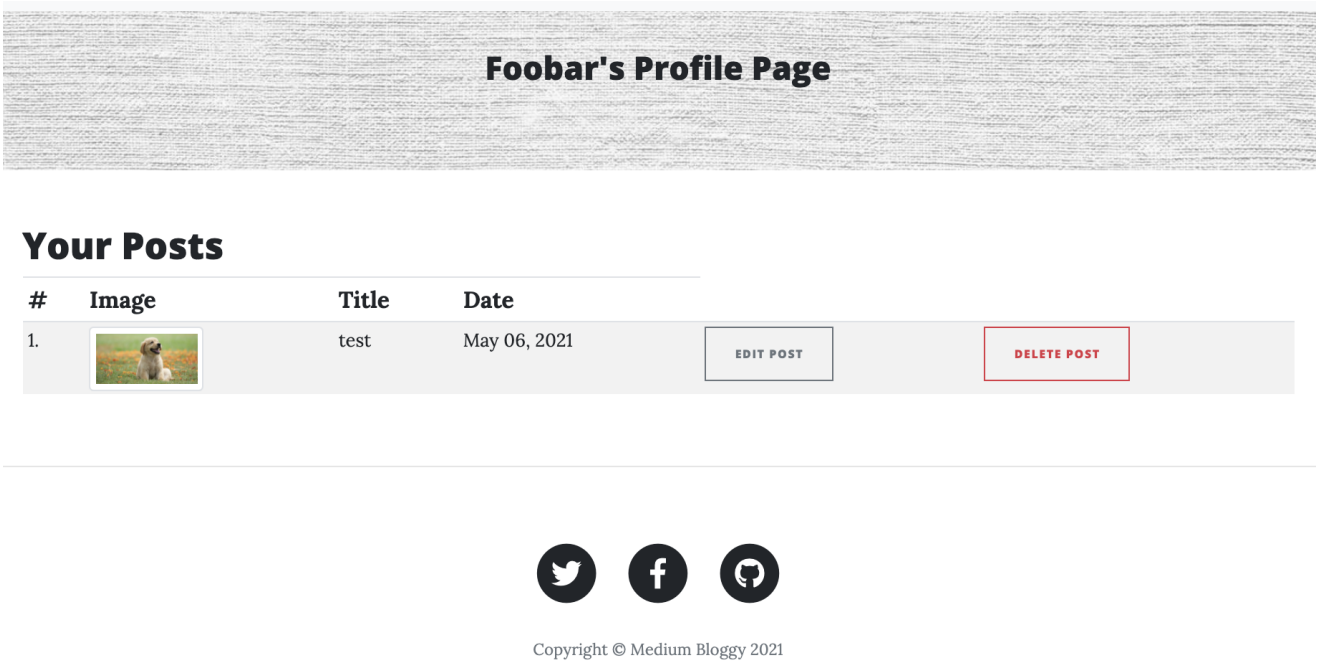


Figure 20: ‘Profile’ dashboard showing all of our Posts. Note the option to ‘Edit Post’ i.e. to Update the Post.

2. Click on ‘Edit Post’ and we are presented with an ‘Edit Post’ form with all of our previously entered data for this particular Post. We now have the option to Edit our Post. Figure 21.

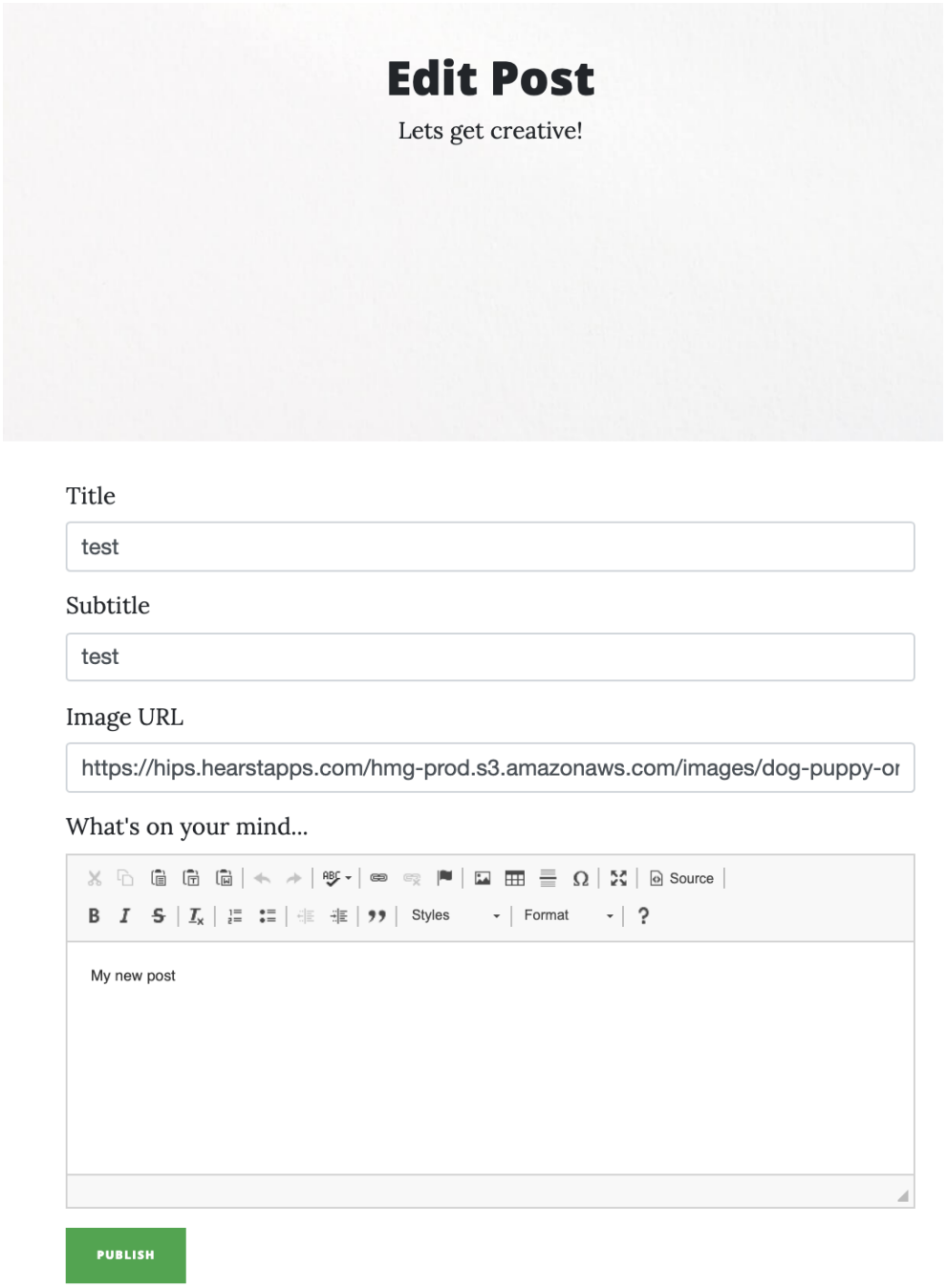


Figure 21: 'Edit Post' screen showing our existing Post details.

3. Once we Edit the Post and click 'Publish' we are redirected back to the 'Read' view of our Post, as shown previously in Figure 18 and 19. Again, if we are the logged in user that created the content, we will also see the option to 'Edit Post' and 'Delete Post'. Figure 22.

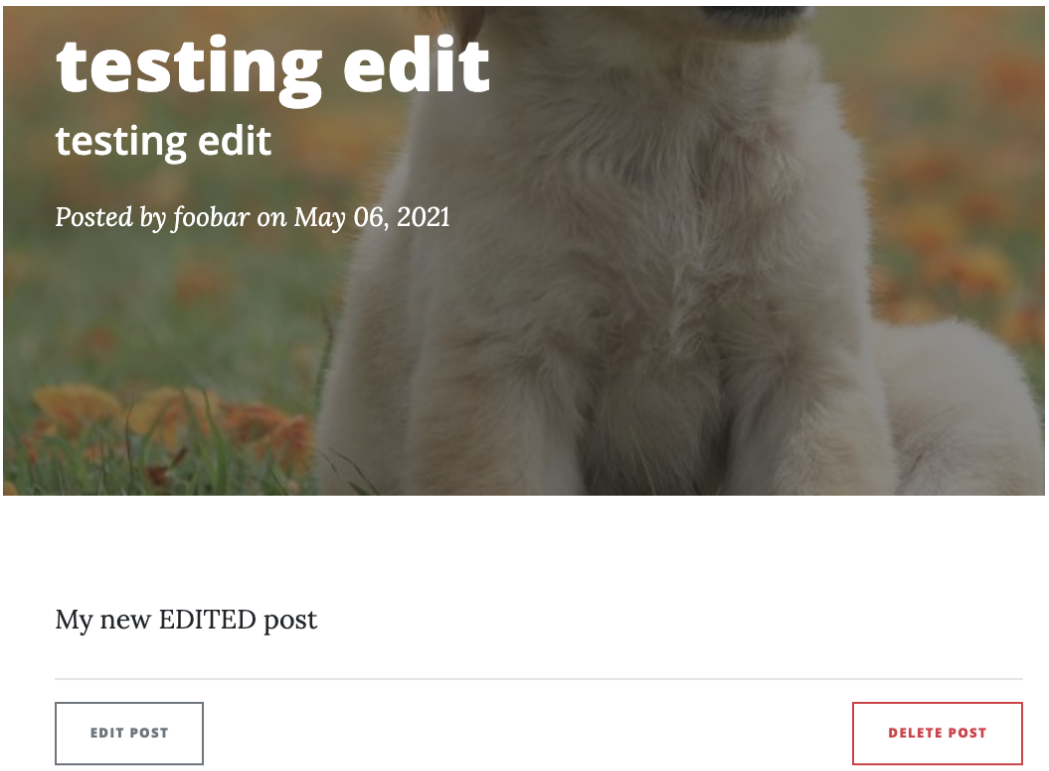


Figure 22: the newly Updated Post.

4. We can confirm the Edit has been successfully transacted on the database. Figure 23.


```
_id: ObjectId("609391172fa3292824ddc1ac")
title: "testing edit"
subtitle: "testing edit"
body: "<p>My new EDITED post<strong> </strong></p>
"
img_url: "https://hips.hearstapps.com/hmg-prod.s3.amazonaws.com/images/dog-puppy..."
author: "foobar"
date: "May 06, 2021"
```


Figure 23: the newly Updated Post, as shown in the database.

## Delete a Post

1. There are two ways to delete a Post - from the Profile page, or from the ‘Read’ version of the Post . If we stay on the ‘Read’ screen from the previous section i.e. Figure 22, we can click on the button ‘DELETE POST’. This brings up the following modal response, where we are asked to confirm if we want to Delete. Figure 24.

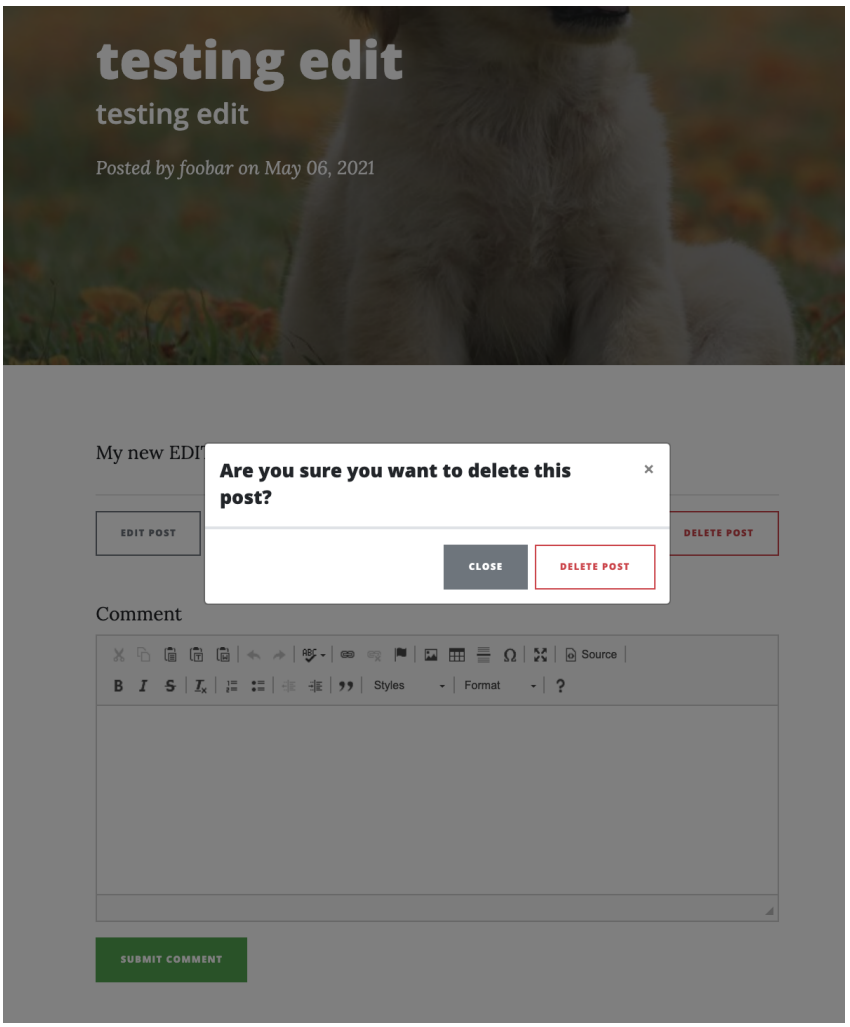


Figure 24: Confirmation modal **before** the Post is deleted.

2. We could also navigate to the 'Profile' page and click on the 'DELETE POST' button. Figure 25.

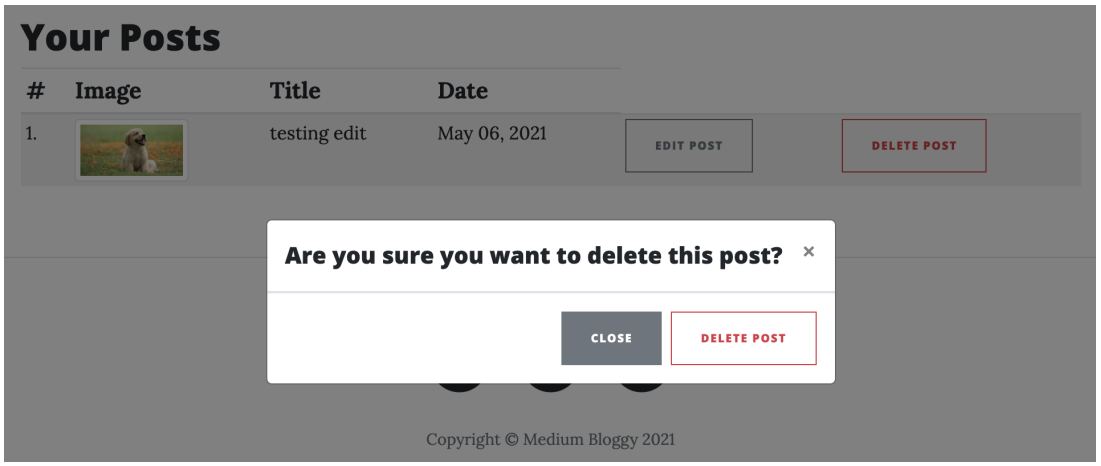


Figure 25: Confirmation modal **before** the Post is deleted, from the 'Profile' page.

3. Once we click ‘DELETE POST’ within the confirmation box, we are redirected back to the main page. The Post is no longer visible. If we navigate to the ‘Profile’ page we can also see that the Post is no longer there, and there is user feedback in the form of ‘Post Successfully Deleted’. Figure 26.

4. We can also confirm that the Post has been deleted from the database by navigating to *‘cloud.mongodb.com’*.

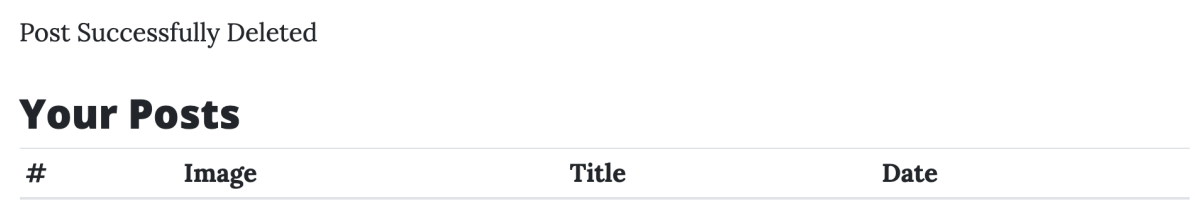


Figure 26: Post has been Deleted from the Profile page, and no longer visible on the main page of the site.

## Testing Create, Read, Delete of Blog Post Comments

### Create a Comment

1. To Create a Comment on a Post, we must be logged in to the site.

2. Once logged in, we can Create a Comment by navigating to the Post, and scrolling towards the bottom of the Post body, where a Form presents itself. Figure 27.

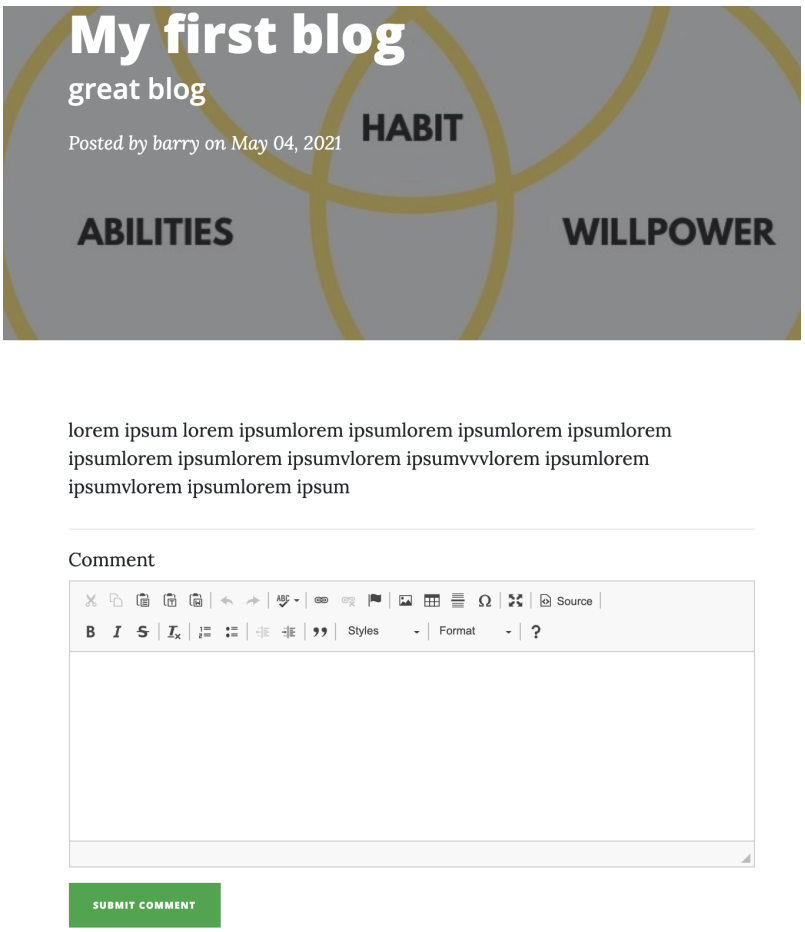


Figure 27: Form for Creating a Comment.

3. Usual validation rules apply, so if we submit an empty Comment, we will receive validation feedback. Figure 28.

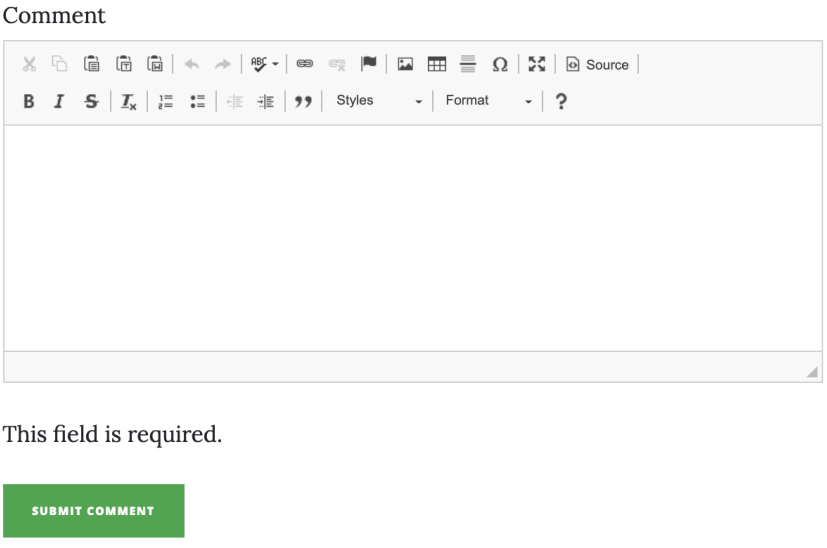


Figure 28: Validation feedback when trying to submit an empty Comment. Note '*This field is required.'* validation comment.

4. Once we have written a valid Comment and clicked the ‘SUBMIT BUTTON’ the page is reloaded and our Comment is immediately visible. Figure 29.

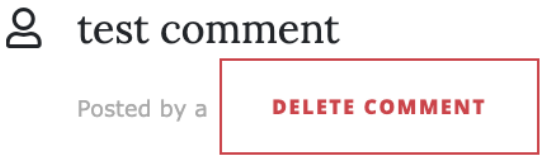


Figure 29: The Comment is now visible. Note the option to ‘DELETE COMMENT’ since the user currently logged in was the creator of the Comment.

5. We can confirm our Comment has been stored in the database by navigating to *‘cloud.mongodb.com’*. Figure 30.

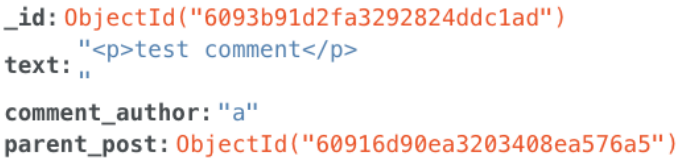


Figure 30: Creation of new Comment in database.

## Read a Comment

1. As a logged in user or as a visitor, clicking on any Post will allow you to Read not only the Post, but also Read any Comments at the end of the Post. Figure 31.

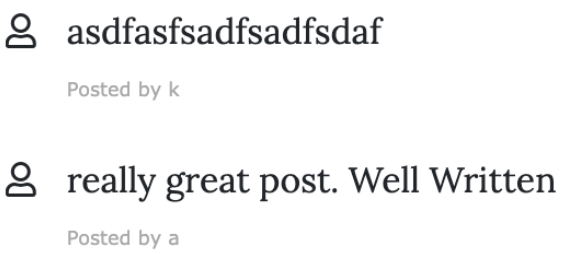


Figure 31: We can ‘READ’ all Comments at the end of the Post.

## Delete a Comment

1. Assuming we are logged in, and as a logged in user we created the Comment, we will also have the option to Delete it. We can navigate to the relevant Post that has our comment, scroll down to where our Comment is located, and click on the ‘DELETE COMMENT’ button. Figure 32.

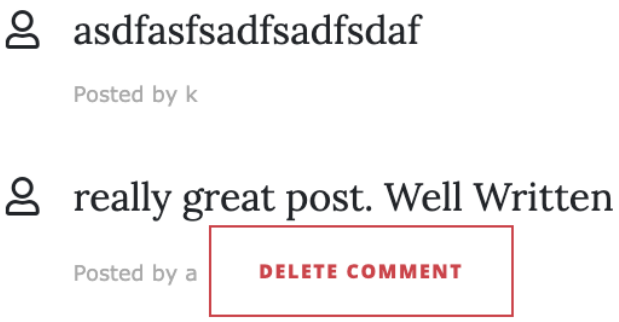


Figure 32: Since user *“a”* is logged in, and since user *“a”* created the *‘really great post. Well Written”* comment, they will also have the option to DELETE COMMENT.

2. Just like Post Deletion, Comment Deletion will pop up a modal confirmation box, for the user to confirm that they want to Delete the Comment. Figure 33.

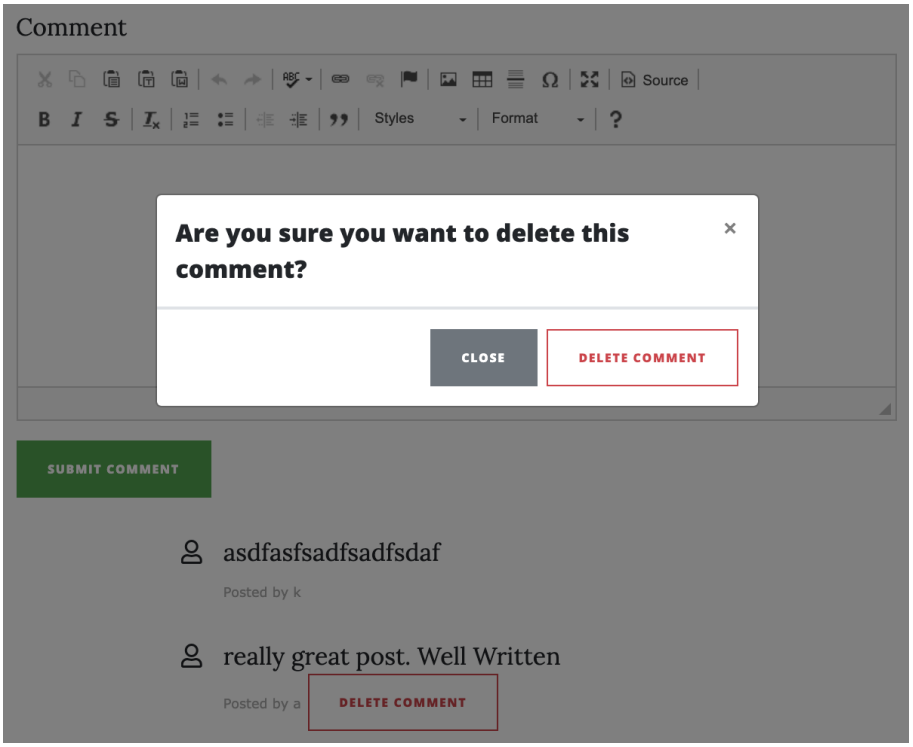


Figure 33: Confirmation modal before the Comment is Deleted

3. Once Deleted, the comment is no longer visible on the Post. Likewise, the Comment has been removed from the database. Figure 34.

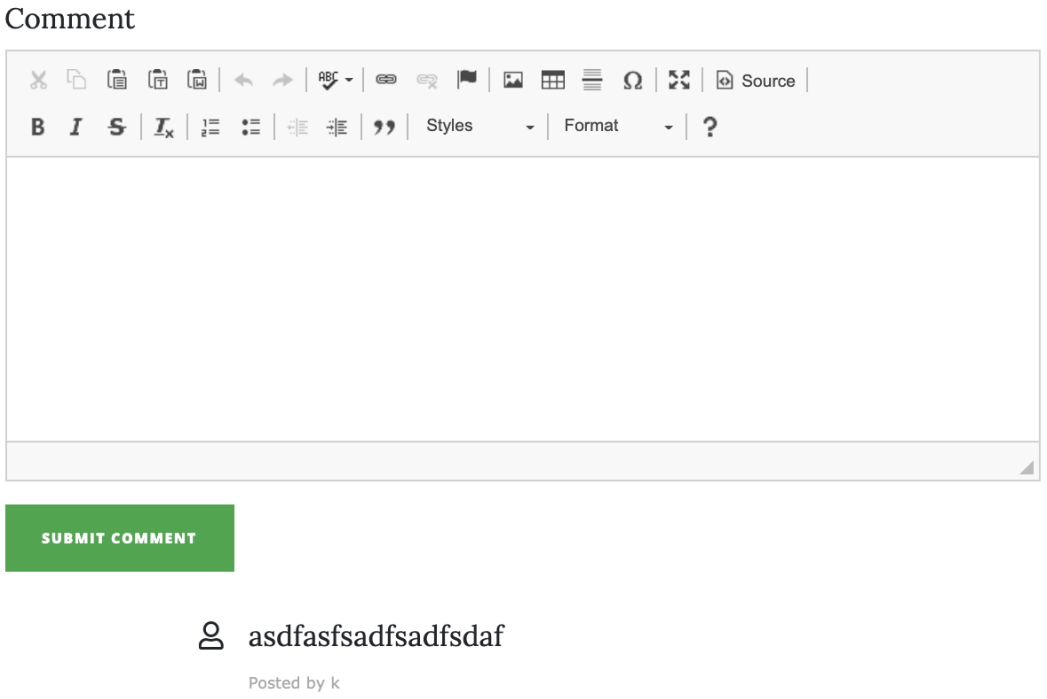


Figure 34: Comment has been Deleted, and is no longer visible.

# 404, 403, 500 Errors

1. We can test 404 Errors by inputting the following URL into our browser:

```
http://mediumbloggy.herokuapp.com/foo
```

2. This results in the following page. Figure 35.

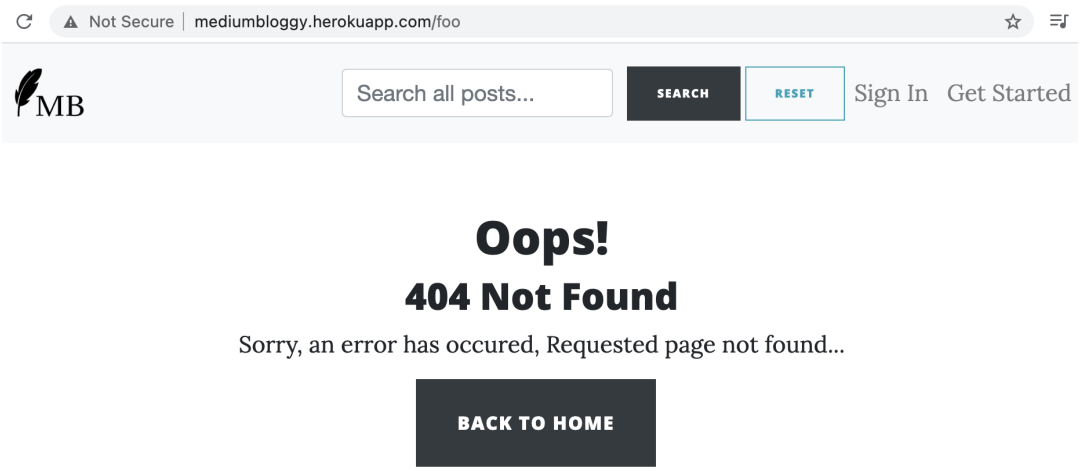


Figure 35: '404 Not Found' template page.

3. We can test 500 Errors by inputting the following URL into our browser:

```
http://mediumbloggy.herokuapp.com/delete/609315e0158d4bee0a03123213
```

4. This results in the following page. Figure 36.

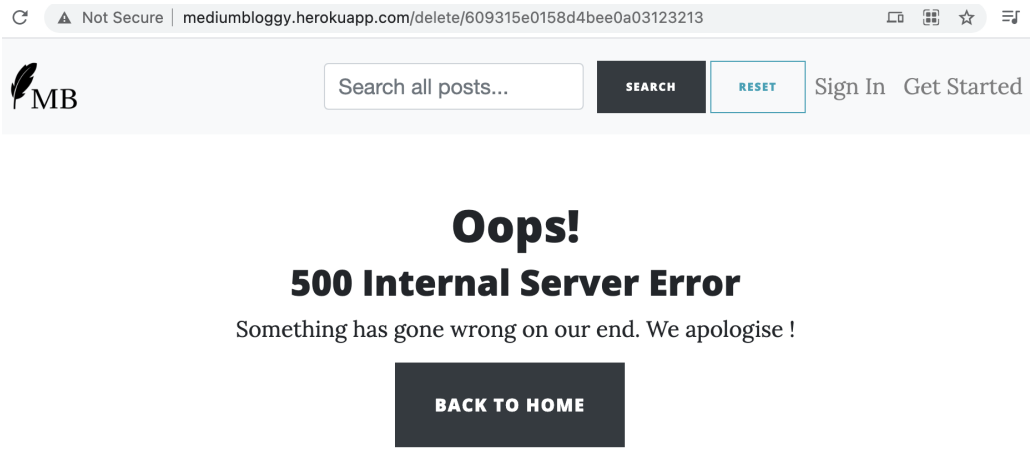


Figure 36: '500 Internal Server Error'' template page.

## Testing URL Protection

URL protection is enforced by ensuring a given URL has a ‘session’. So, for example:

1. The URL for creating a post is:

```
http://mediumbloggy.herokuapp.com/create-post
```

2. If a user that is not logged in tries to access this URL, they will be **redirected** back to the login page i.e.

```
http://mediumbloggy.herokuapp.com/login
```

3. This ensures that our URLs are protected from users that do not have a current ‘session’.

## Testing Search Functionality

1. We can test the Search functionality by typing in a relevant search query into the ‘Search all posts...’ input field on the navigation bar. Figure 37.

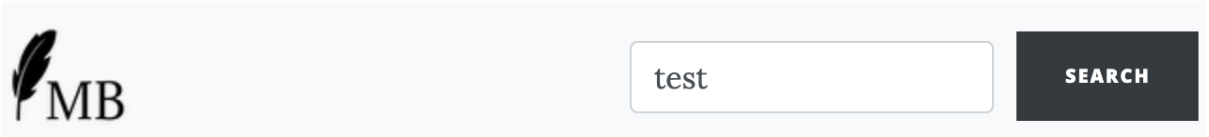


Figure 37: searching for any Post with the word ‘test’ in its Title or Subtitle.

2. After clicking SEARCH we can see that the relevant Post is returned. Figure 38.

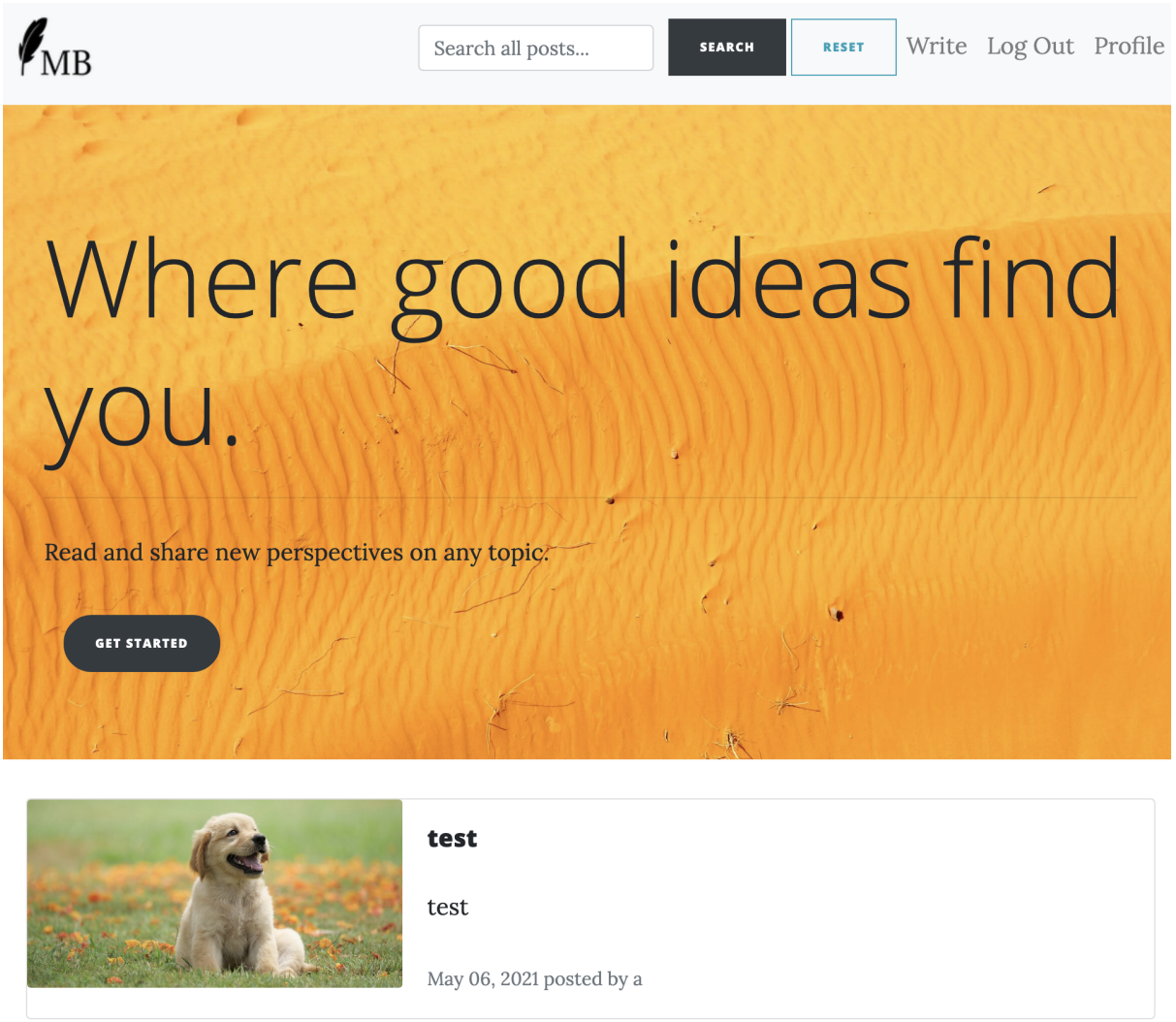


Figure 38: the relevant Post is returned.

3. We can also test what would happen if nothing was returned. We can type in '*dog'* into the 'Search all posts...' input field. We receive the following output. Figure 39.


Figure 39: Searching for *'dog'* as a Title or Subtitle returned a 'No results Found' message.