

# Microblog

Developed by Debmalya Jash

<b>Purpose</b>	<b>2</b>
<b>Github URL</b>	<b>2</b>
<b>Technology Stack</b>	<b>2</b>
<b>Responsive Web Design</b>	<b>3</b>
Functionality / User Journey	3
Login	3
User Registration	4
Listing all blogs	4
Functionality Search blogs	6
Functionality get list of users	7
Create a simple post	9
View individual post	10
Update a simple blog	10
Display of Status Message	12
Delete a simple post	13
Display of Error Message	14
Menu	16
List of APIs	16
List of views (twig templates)	18
List of javascripts	18
<b>API Web Service Changes</b>	<b>19</b>
Functionality	19
Login validation	19
Listing all posts	19
Create User table	19
Get list of users (Added functionality)	20
Create a simple blog	20
View individual blog	20
Delete a simple blog	21
Update a simple blog	21
Having Fun	21
Error handling	22
Validation	22
User Session Management	23
Performance	23

Developer : Debmalya Jash

File List	23
Before release checklist	24
Improvement	26
Lazy fetching and pagination.	26
Reduce communication with server. Communication with server should be as minimum as possible.	26
Service /Template Controller	26
Reduce database communication	26
Add test code	27
Twig macro	27
Prevent SQL injection	27
UTF-8 character handling	27
UI error checking.	27
Security	27

## Purpose

Thanks for providing this opportunity to develop this micro-blog sample application. This document is created to describe the changes made for micro blog development.

## Github URL

<https://github.com/debmalya/micro-blog-technical-test.git>

## Technology Stack

W3-CSS
W3-data
jQuery
Silex (1.3)
Twig (1.2 )
PHP 5.6.24
doctrine/dbal (2.2)
SQLite

Developer : Debmalya Jash

Nginx
Cent OS' 7

## Responsive Web Design

Functionality / User Journey

Login

# Micro Blog

User registration successful. Please login to continue.

Username

Password

Register

Login

Developer: Debmalya Jash

Existing user can login using this page. New user can register.

If there is no users table. Application creates users table with default values

Username	Password
User1	User1
User2	User2
User3	User3
User4	User4

Developer : Debmalya Jash

## User Registration

# Micro Blog

Register

Username

Password

Register

Developer: Debmalya Jash

This is for user registration. Confirm-password is missing now. Later I will include this one.

On successful user registration user will be directed to login page.

# Micro Blog

User registration successful. Please login to continue.

Username

Password

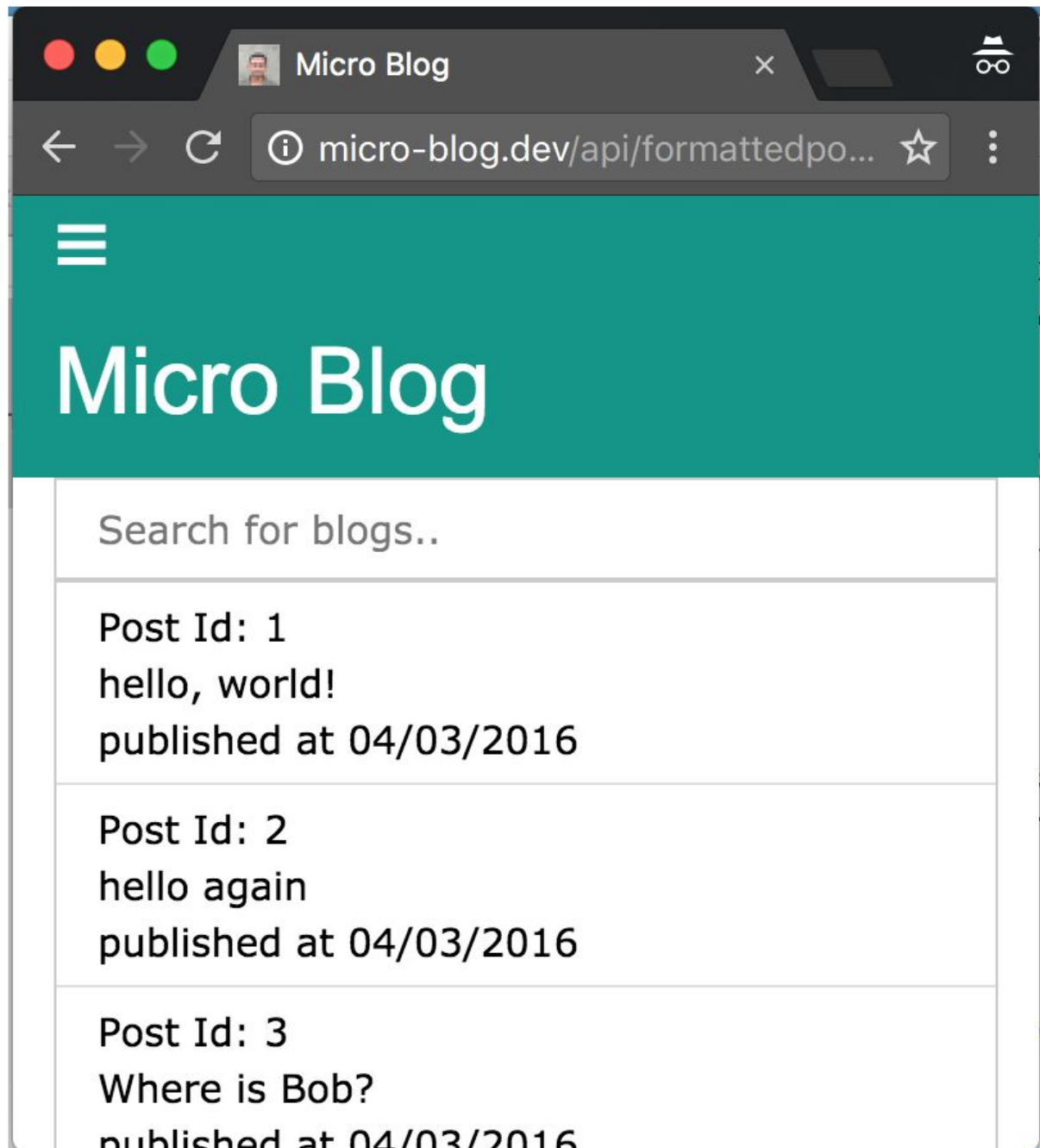
Register

Login

Developer: Debmalya Jash

## Listing all blogs

Developer : Debmalya Jash



Blogs.twig has following contents to display the blog posts.

```
{% if posts|length > 0 %}
```

```
<ul id="id01" class="w3-ul w3-border">
```

```
{% for post in posts %}
```

```

<li>Post Id:
{{post.rowid}} <br> {{ post.content|nl2br }} <br> published at {{ post.date is empty
? "" : post.date|date("d/m/Y","Europe/London") }}</li>

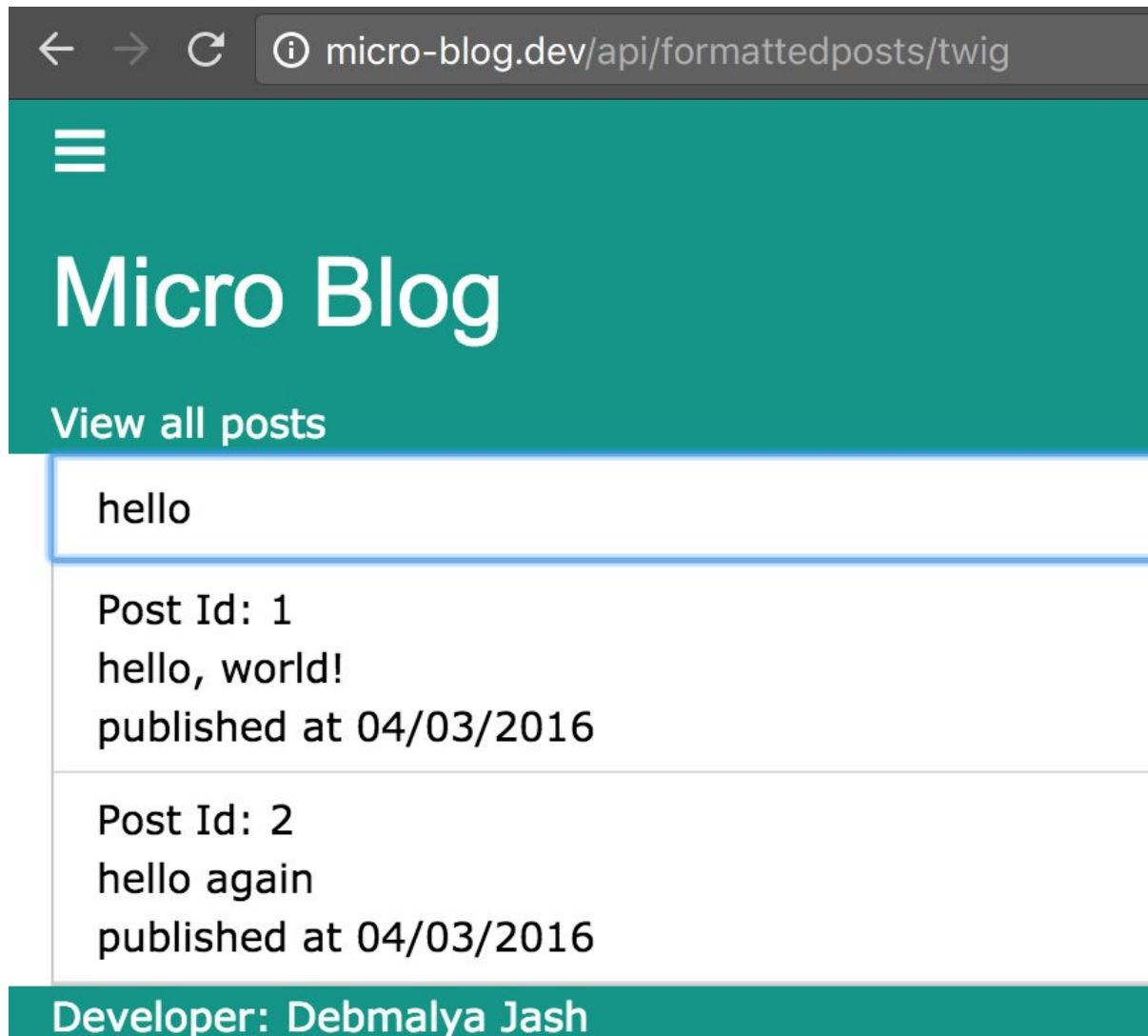
{% endfor %}

</ul>

{% endif %}

```

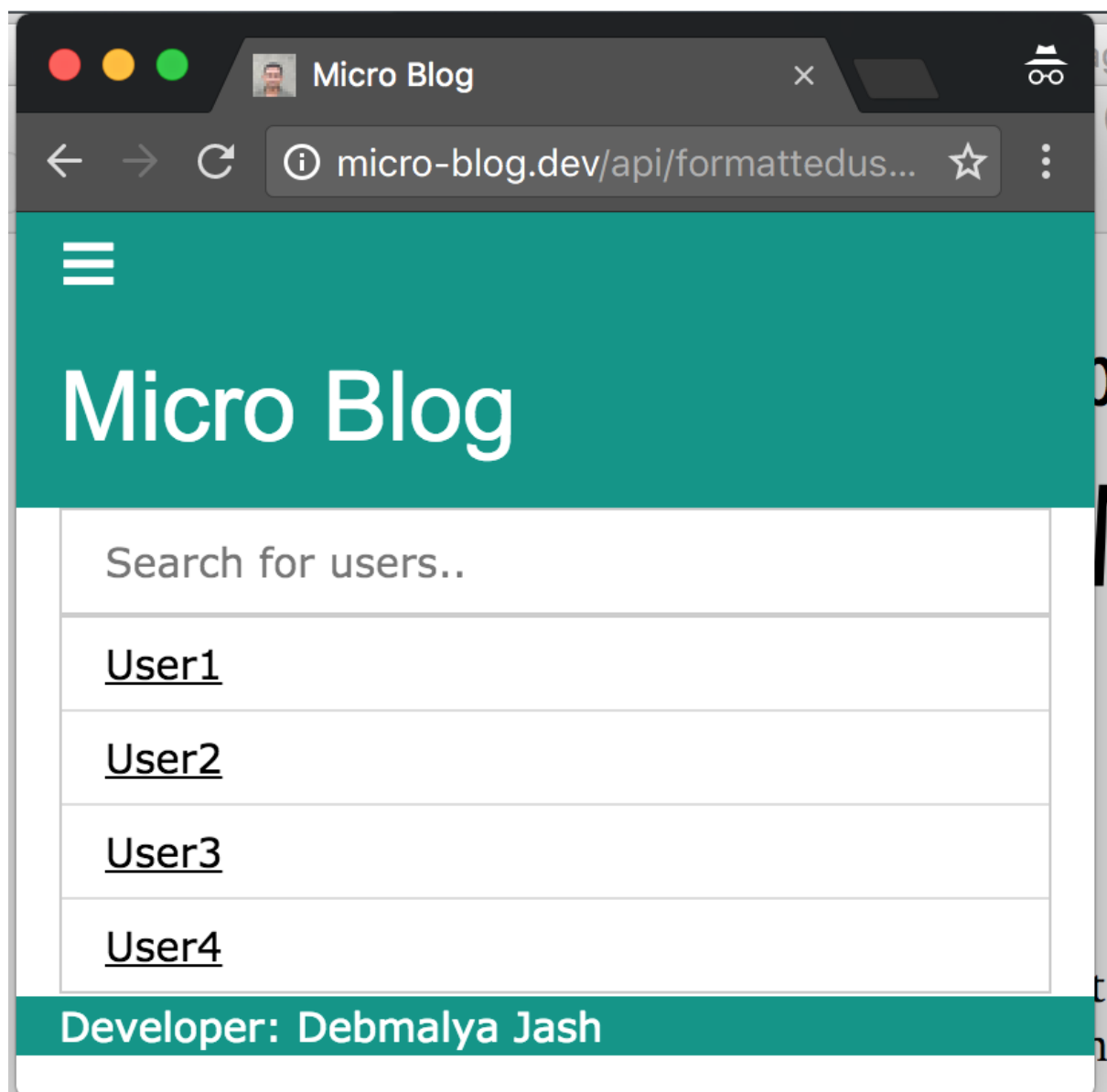
Functionality Search blogs



Developer : Debmalya Jash

This search is case insensitive. I am using w3 css / data libraries for this.

Functionality get list of users

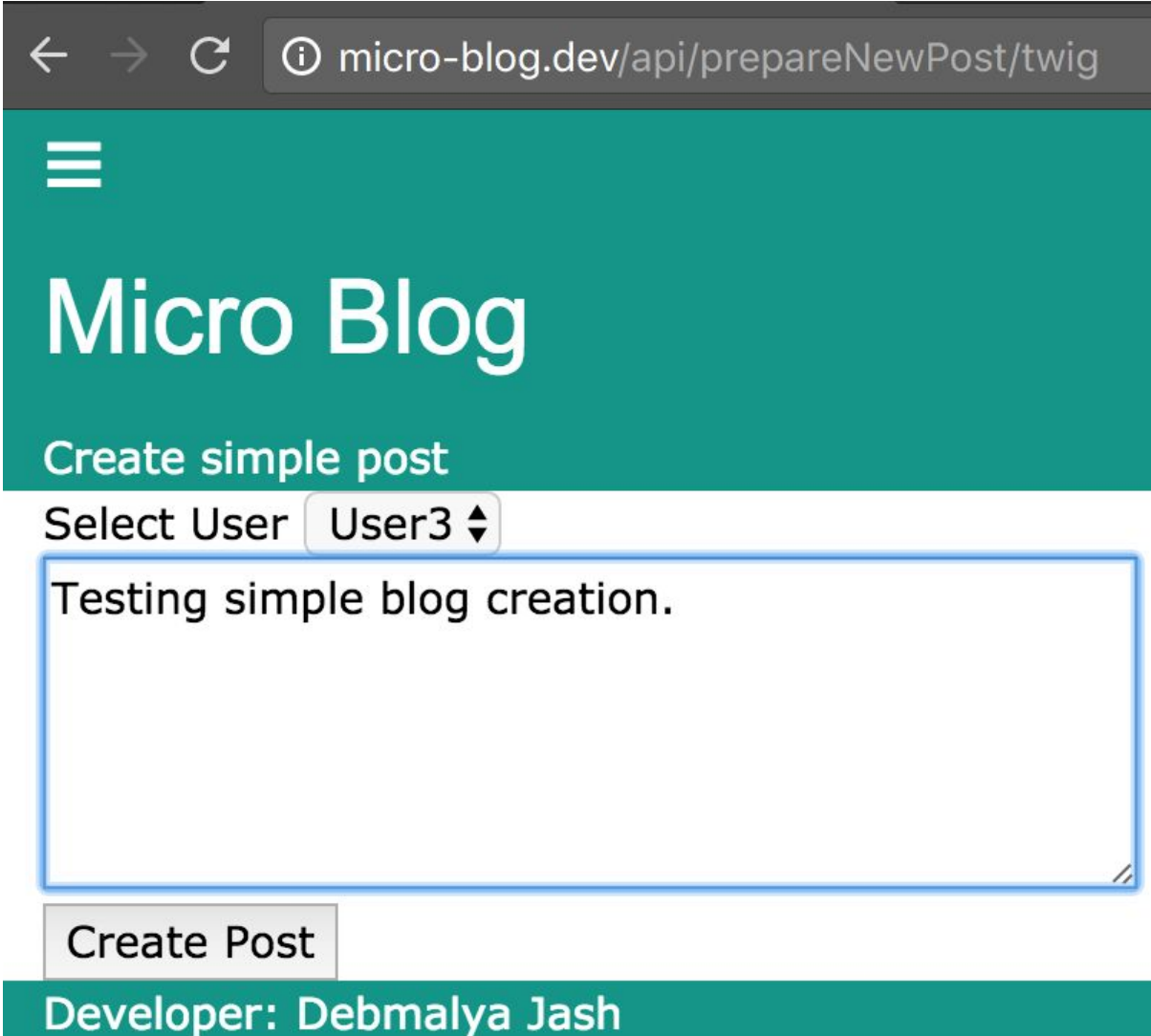


Developer : Debmalya Jash

Each user will be a link. On clicking on that user it will display the list of blogs posted by that user.



Create a simple post



The screenshot shows a web browser window with the address bar displaying `micro-blog.dev/api/prepareNewPost/twig`. The page has a teal header with a hamburger menu icon and the title "Micro Blog". Below the header, the text "Create simple post" is displayed. A "Select User" dropdown menu is set to "User3". A large text input field contains the text "Testing simple blog creation.". Below the input field is a "Create Post" button. The footer of the page, also in teal, displays "Developer: Debmalya Jash".

← → ↻ ⓘ micro-blog.dev/api/prepareNewPost/twig

☰

# Micro Blog

Create simple post

Select User User3 ▾

Testing simple blog creation.

Create Post

Developer: Debmalya Jash

Developer : Debmalya Jash



# Micro Blog

Blog created successfully.

Welcome !! Click on menu to do different activities.

Developer: Debmalya Jash

View individual post

The screenshot shows the 'Micro Blog' application interface. At the top, there is a teal header with a menu icon and the title 'Micro Blog'. Below the header, a teal bar displays 'View your individual blog , User1'. The main content area is a white table with a search bar at the top. The table contains two rows of blog posts. The first row shows 'Post Id: 5', 'My first blog.', and 'published at 17/10/2016'. The second row shows 'Post Id: 7', 'Enter your blog here', and 'published at 17/10/2016'. At the bottom of the table, a teal bar displays 'Developer: Debmalya Jash'.

Search for blogs..		
Post Id: 5	My first blog.	published at 17/10/2016
Post Id: 7	Enter your blog here	published at 17/10/2016

Developer: Debmalya Jash

This will list all the blogs of currently logged in user.

Update a simple blog

From the menu select 'update a simple blog'.

From the drop down select postid.

Developer : Debmalya Jash



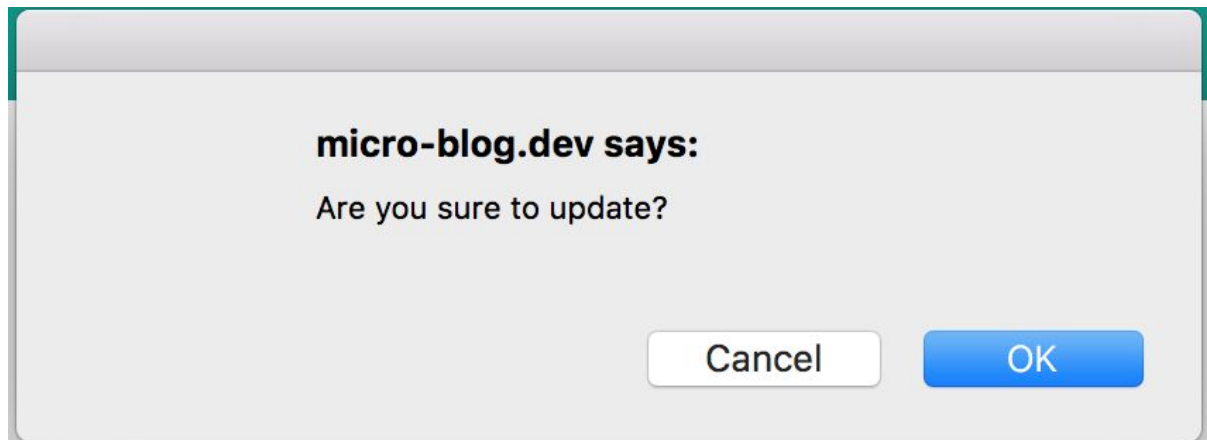
The screenshot shows a web application titled "Micro Blog" with a teal header. Below the header, the text "Update a simple post" is displayed. A dropdown menu labeled "Select Post" is open, showing two options: "5" (selected with a checkmark) and "7". Below the dropdown is a large text area containing the text "My first blog. Updating now". At the bottom of the text area is a small double-slash icon. Below the text area is a button labeled "Update Post". The footer of the application is teal and contains the text "Developer: Debmalya Jash".

Here in the drop down the postid will appear. These postid s are created by the currently logged in user.

Known issue: If there is only post, then on clicking this drop down then blog entry is not changing. This is due to in jQuery i have used drop down 'change' event. If there is only one on clicking there is no change in value in the text area.

Modify the post.

It will ask for confirmation  
Please Click on 'Update Post'.



Display of Status Message

Status message "Blog updated successfully".



Developer : Debmalya Jash

Delete a simple post



No post available

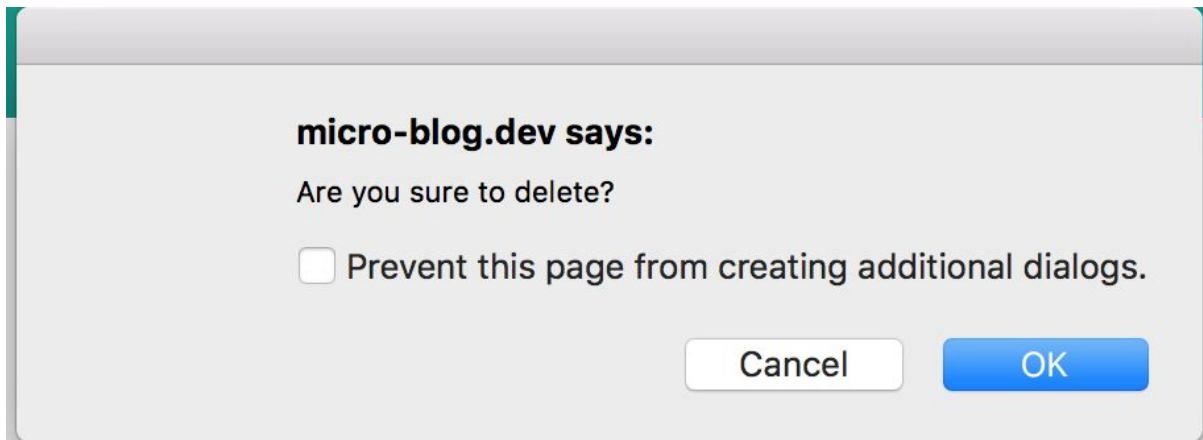
Developer: Debmalya Jash

For this user currently there is no post to delete. If there are any post, it will appear like below.



Developer : Debmalya Jash

On clicking 'Delete Post' following confirmation box will appear.



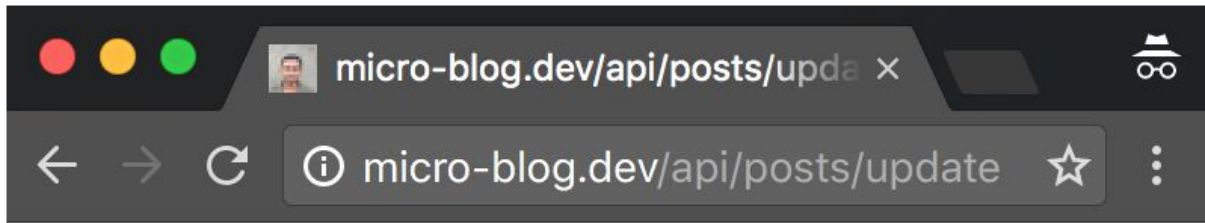
On confirming blog will be deleted. Following status message will appear.



Here 'Raju' is the user name.

Display of Error Message

Sample error message screen.



I am sorry, but something went terribly wrong. Error Code :500 Error message :An exception occurred while executing 'UPDATE posts SET content = :content WHERE rowid = :rowid' with params [":content", "hello again", ":rowid", 2]: SQLSTATE[HY000]: General error: 25 bind or column index out of range

I will try to represent the errors in prettier way.

## Menu

Close ×

View all posts

View all users

View an individual post

Create a simple post

Update a simple post

Delete a simple post

Logout

## List of APIs

In general API name containing 'formattedXXX' is for application usage. Other API are for normal usage then return JSON response.

API	Status	Description	Remarks
/api/posts	Modified	Existing	Provided checking if there is no post at all.
/api/formattedposts/{format}	New	To get posts in different format. Currently It supports	

Developer : Debmalya Jash



		twig template.	
/api/formattedusers/{format}	New	To get user details in different formats. Currently it supports twig template.	
/api/posts/user/{user_id}	Modified	Existing	Provided checking if there is any post for the user.
/api/posts/formatteduser/{user_id}/{format}	New	To get all the blogs of an user in different formats. Currently it supports twig template.	Each user has hyperlink. On clicking on the hyperlink all the blogs for that user will appear.
/api/posts/id/{post_id}	Modified	Existing	Provided checking if there is any post for the post id.
/api/formattedposts/{format}/{post_id}	New	To get post for the passed id in different format . Currently it supports twig template.	
/api/posts/new	New	To create new post	
/api/formattedposts/new	New	To create new post get specific template, currently supported template is 'twig'.	
/api/prepareNewPost/{format}	New	To prepare template for create simple blog. It will populate the list of users.	
/api/posts/delete	New	To delete a post	
/api/posts/update	New	To update a post	
/api/postid	New	To get all rowids from posts	
/api/postid/{id}	New	To get content of a blog. Id is passed as parameter.	

/api/login/validate	New	To validate user	
/api/formattedlogin/validate	New	To validate user and return to the main menu of the application	

### List of views (twig templates)

Twig template name	Purpose	Remarks
blogs.twig	To display blogs	Try to merge these three into one. Based on some parameter either delete, update link button will appear.
create.post.twig	To create blog	
single_blog.view.twig	To view individual blog	
single_blog.update.twig	To update individual blog	
single_blog.delete.twig	To delete individual blog	
footer.twig	To display footer	
form.html	Trying with twig macro	
header.twig	To display header	
index.twig	To display the landing page	
user.link.twig	To display user. Each user is a link to display all of their blogs.	
users.twig	To display users without any link.	

### List of javascripts

Javascript file name	Purpose	Remarks
jquery-3.1.1.min.js	jQuery library	To call web service, handling different event like form confirmation, drop down selection.
functionality1.js	To handle menu opening	

Developer : Debmalya Jash

	and closing.	
w3data14.js	W3data javascript library	W3 CSS and W3 data library used for responsive web designing. They are reported to be faster than bootstrap.
blog_controller15.js	With the help of jQuery and w3data14.js here I handled web service call, form confirmation, on keyup event user search and blog search.	In future like to make it a full fledged controller. More separation of code from model.

## API Web Service Changes

### Functionality

#### Login validation

#### Listing all posts

##### Changes in index.php

```
$app->get('/api/posts', function() use($app) {
    $sql = "SELECT rowid, * FROM posts";
    $posts = $app['db']->fetchAll($sql);

    if (count($posts) == 0) {
        return new Response("There is no post.", 404);
    }

    return $app['twig']->render('blogs.twig', array('posts' => $posts, ));
});
```

#### Create User table

```
use Doctrine\DBAL\Schema\Table;
```

```
// Create user table
```

```
$schema = $app['db']->getSchemaManager();
if (!$schema->tablesExist('users')) {
    $users = new Table('users');
    $users->addColumn('user_id', 'integer', array('unsigned' => true,
'autoincrement' => true));
```

Developer : Debmalya Jash

```

$users->setPrimaryKey(array('user_id'));
$users->addColumn('user_name', 'string', array('length' => 32));
$users->addUniqueIndex(array('user_name'));
$schema->createTable($users);

// insert sample rows
$app['db']->insert('users', array( 'user_name' => 'User1',));
$app['db']->insert('users', array( 'user_name' => 'User2',));
$app['db']->insert('users', array( 'user_name' => 'User3',));
$app['db']->insert('users', array( 'user_name' => 'User4',));
}

```

### Get list of users (Added functionality)

```

$app->get('/api/users', function() use($app) {
    $sql = "SELECT rowid, * FROM users";
    $posts = $app['db']->fetchAll($sql);

    if (count($posts) == 0) {
        return new Response("There is no users.", 404);
    }

    return $app['twig']->render('users.twig', array('posts' => $posts, ));
});

```

### Create a simple blog

```

$app->post('/api/posts/new', function (Request $request) use ($app) {
    $user_id = $request->request->get('user_id');
    $content = $request->request->get('content');

    $app['db']->insert('posts', array( 'content' => $content, 'user_id' =>
(int)$user_id, 'date' => time()));
    $posts = array('message' => 'Blog created successfully. ');
    return $app->json($posts, 200);
});

```

Where is error handling ? - I will try to put this in try catch block.

Where is retry mechanism? If there is database down, or network issue will it retry ?

### View individual blog

To view individual blog i rely on following apis

/api/postid

/api/formatted/postid/{format}

## Delete a simple blog

```
$app->delete('/api/posts/delete', function (Request $request) use ($app){
    $post_id = $request->request->get('post_id');
    $app['db']->delete('posts', array( 'rowid' => (int)$post_id,));
    $posts = array('message' => 'Blog deleted successfully.','rowid'=>$post_id);
    return $app->json($posts, 200);
});
```

Needs more error handling like database down, network issue etc. Have to implement retry mechanism (delayed retry, count retry (e.g. 3 times).

## Update a simple blog

```
$app->put('/api/posts/update', function (Request $request) use ($app){
    $post_id = $request->request->get('post_id');
    $content = $request->request->get('content');
    $sql = "UPDATE posts SET content = :content WHERE rowid = :rowid";
    $app['db']->executeUpdate($sql, array($content,(int)$post_id));
    $posts = array('message' => 'Blog id $post_id updated successfully.');
```

```
    return $app->json($posts, 200);
});
```

## Having Fun

Making new API is fun. Let me have more fun. I want to have an api, which will give me the posts from particular range (rowid 0 to rowid 4). How will it look like?

```
$app->get('/api/posts/range/{from}/{to}', function($from,$to) use($app) {
    $sql = "SELECT rowid, * FROM posts where rowid >= ? and rowid <= ?";
    $posts = $app['db']->fetchAll($sql, array((int) $from,(int) $to));
    if (count($posts) == 0) {
        return new Response("There is no record.", 404);
    }

    return $app->json($posts, 200);
});
```



The screenshot shows a web browser address bar with the URL `micro-blog.dev/api/posts/range/0/4`. The response body is a JSON array of four objects, each representing a blog post with fields: `rowid`, `content`, `user_id`, and `date`.

```
[{"rowid": "1", "content": "hello, world!", "user_id": "0", "date": "1457084104"}, {"rowid": "2", "content": "hello again", "user_id": "0", "date": "1457084404"}, {"rowid": "4", "content": "Hi, I\\u0027m Charlie", "user_id": "2", "date": "1457083821"}]
```

```
/**
 * Get total number of posts.
 */
$app->get('/api/posts/total', function() use($app) {
    $sql = "SELECT count(*) FROM posts";
    $posts = $app['db']->fetchAll($sql);
    if (count($posts) == 0) {
        return new Response("There is no post.", 404);
    }
});
```

Developer : Debmalya Jash

```
return $app->json($posts, 200);
});
```



```
[ { "count(*)" : "12" } ]
```

```
/**
 * Get total number of posts group by user
 */
$app->get('/api/posts/group_by_user', function() use($app) {
    $sql = "SELECT user_id,count(*) FROM posts group by user_id";
    $posts = $app['db']->fetchAll($sql);
    if (count($posts) == 0) {
        return new Response("There is no post.", 404);
    }

    return $app->json($posts, 200);
});
```

```
[{"user_id":"0","count(*)":"2"}, {"user_id":"1","count(*)":"2"}, {"user_id":"2","count(*)":"2"}, {"user_id":"3","count(*)":"3"}, {"user_id":"4","count(*)":"3"}]
```

### Error handling

```
$app->error(function (\Exception $e, $code) {
    return new Response('I am sorry, but something went terribly wrong.' . " Error Code : " . $code . " Error message : " . $e->getMessage());
});
```

### Validation

1. Post id must be a positive integer, ->assert('id', '\d+');
2. Error message for non existing post id.



Post id 10 does not exist.

3. Error message for non existing user id.



User id 9 does not exist.

## User Session Management

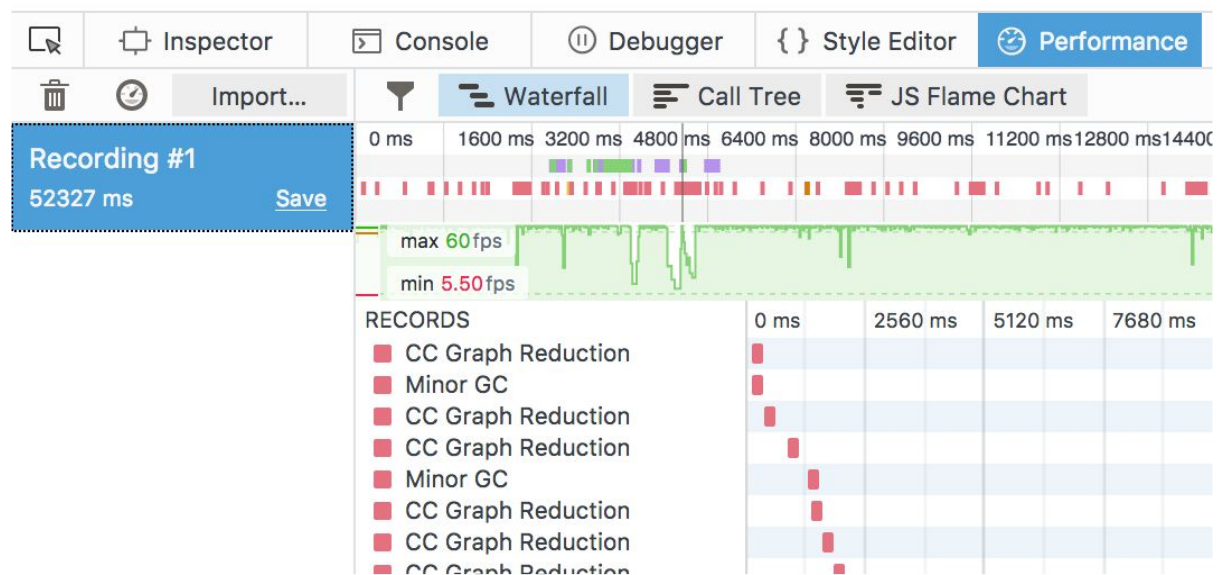
On successful login, storing user\_name and user\_id in session.

```
$app['session']->set('user', array('user_name' =>
$user_name, 'user_id' => $posts['user_id']));
```

Stored values are cleared during logout operation

```
$app['session']->clear();
```

## Performance



Will test more.

## File List

File Name	Location	Purpose
index.php	src	Our model.
All javascripts files	src/assets/javascripts	Managing the show.

Developer : Debmalya Jash

All twig files	src/views	The view of three musketeers MVC.
----------------	-----------	-----------------------------------

### Before release checklist

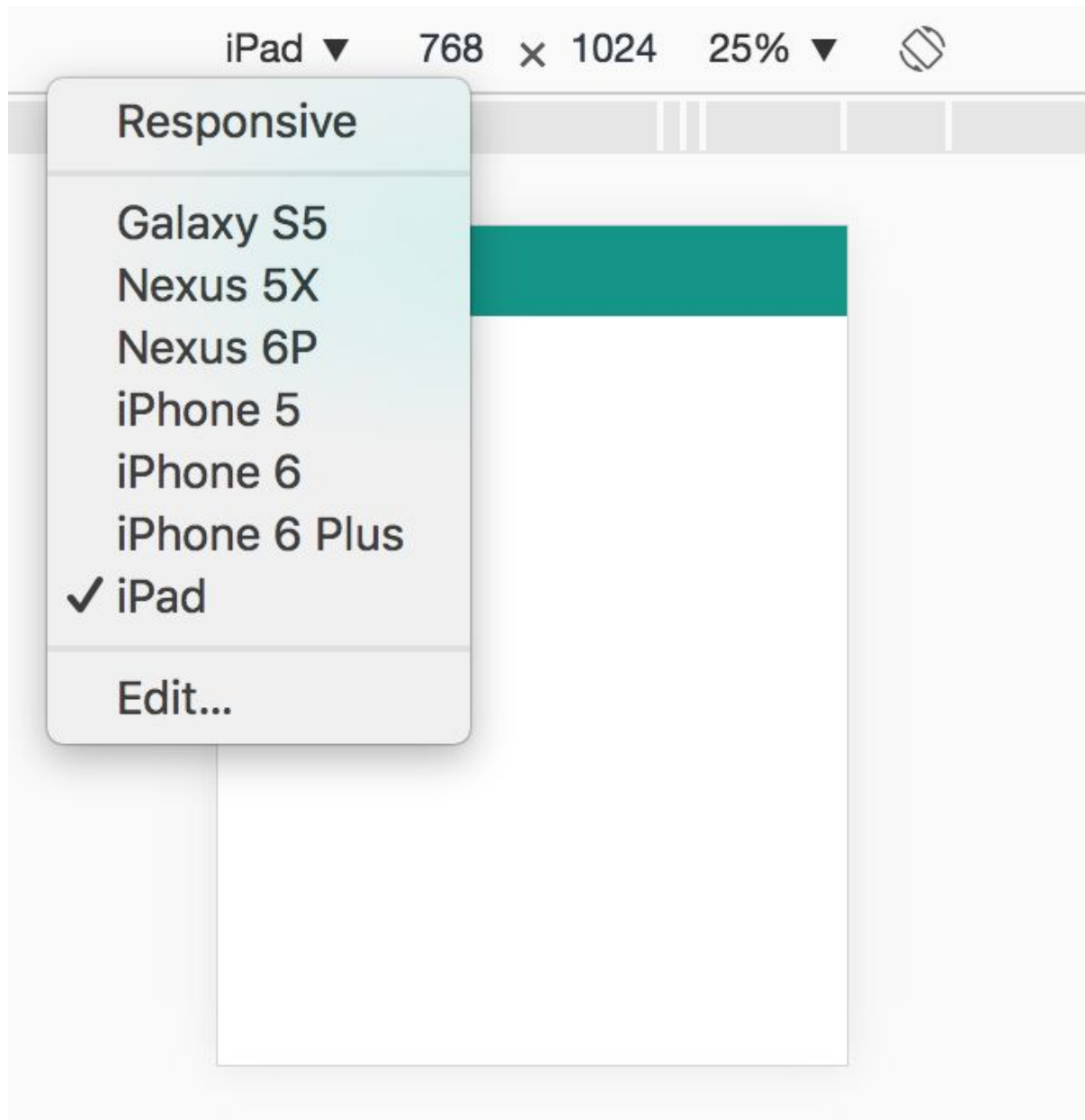
Functionality	OK ?
1. View all posts?	
2. View all Users?	
3. After getting all the list of users, click on the user link to check whether all his/her post appeared?	
4. Whether view individual post is working or not?	
5. Check whether create a simple post is working or not?	
6. Check whether update a simple post is working or not?	
7. Check whether delete a simple post is working or not?	

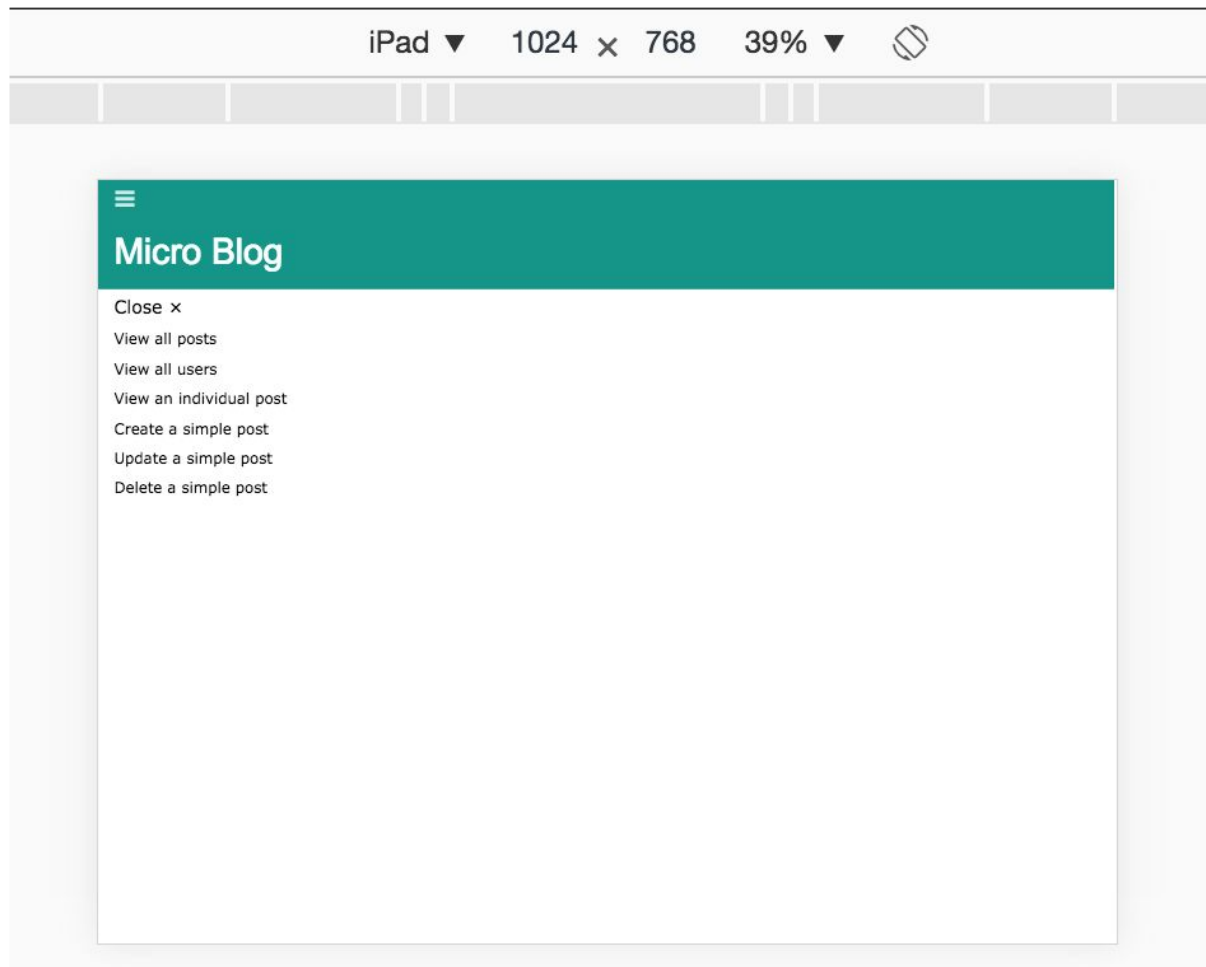
Check these functionalities in the following browser

Browser Name	OK?
IE	
Google Chrome	
Safari	
Firefox	

Check these functionalities in different screen resolution







## Improvement

Lazy fetching and pagination.

Reduce communication with server. Communication with server should be as minimum as possible.

Service /Template Controller

Create a template controller or template engine, so that template provider (e.g. twig) can be changed any time. Template provider change will be just a matter of configuration and addition /modification of template files.

Reduce database communication

Use [HttpCacheServiceProvider](#) to reduce database hit.

Developer : Debmalya Jash

Add test code

For this I can use [phpunit](#) or Web test case.

Twig macro

I can improve twig files by create a macro for the basic one, then extending on that.

Prevent SQL injection

Have to use `htmlspecialchars`

UTF-8 character handling

This one is not tested.

UI error checking.

There should be more error checking like for

- Empty string.
- Exceeding max length.
- Checking for special character.

Security

Password is not stored in encrypted way. `password_hash()` or `crypt()` function can be used.