Coursework 2 Assignment Report

Natalia Czekalska
40286736@live.napier.ac.uk
Edinburgh Napier University - Web Technologies (SET08101)

Abstract

This report describes the process of implementation of a website providing CRUD features (Create, Read, Update, Delete). This is four basic functions in applications that are using non-volatile memory. These applications are using a database to store the information and each function can be accomplished through a database query (POST, PUT, DELETE and GET in HTTP environment).

1 Introduction

This coursework required to design, implement and evaluate a blog platform. The implementation of this blog platform has to include both server and client elements. The client element will present a user interface enabling at least one user to add a new blog post, to edit or view an existing blog post, and to delete an existing blog post. The server element will persist data related to the blog, will serve up the user interface, and will also provide a create, read, update, delete function. Technologies that must be used are HTML, CSS and Javascript for the client interface and Node.JS on the server.

I have decided to create a basic one page web application using MEAN stack. MEAN stands for MongoDB, express, AngularJS and NodeJS.

- **MongoDB** is a free and open-source, document-oriented, NoSQL database that uses JSON-like documents with schemas.
- **Express** is a minimal and flexible Node.JS web application framework for building single and multi-page web applications.
- **AngularJS** extends HTML vocabulary for the application resulting in expressive, readable, and quick to develop environment.
- **NodeJS** is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications.

2 Software design

Project requirements based on the coursework brief:

- The web application has to allow to add , delete, read and edit posts
- The application should persist the data
- It should have solid design and provide at least acceptable user experience
- The colours and fonts chosen should make the text easy to read.
- Functionality should be consistent with a cohesive overall design.

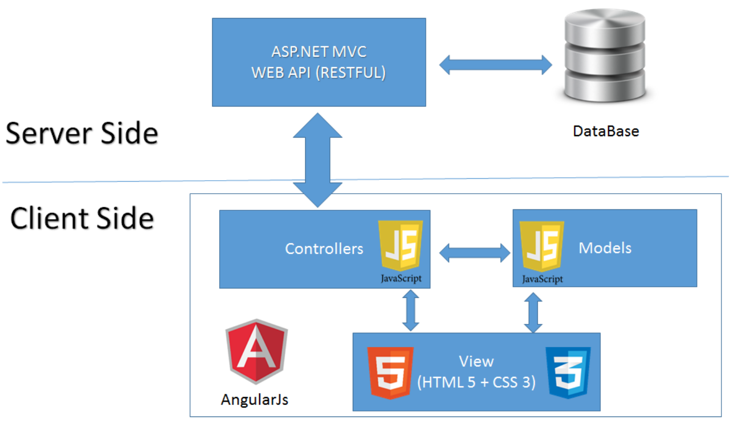
Based on the requirements I plan to implement:

- **CRUD** Basic functions allowing a user to add, delete, view and update a post,
- **Sorting** One of the blog features is that the posts are displayed from the newest to the oldest,
- **Register/Login** If there will be enough time left I will try to implement support for multiple users

I have started my work on this project by doing research on MongoDB database and creating my account on https://mlab.com/. Because I have decided to work on a MEAN stack application I had to install the necessary components like express and nodeJS. After that I went through many tutorials available online about building MEAN web apps. The 3 I have completed were *Creating a Simple Blog Using MEAN Stack* [3] , *Learn to Build Modern Web Apps with MEAN* [6], and *MEAN Stack Tutorial with Angular 2* [1].

After I completed these tutorials I decided that my page will be built based on a Model View Controller architecture. It means that the data will be separated between presentation (View), data (Model) and action (Controllers).

Figure 1: MVC website logic *http://www.codeproject.com/KB/aspnet/1068088/architectureA.png*



3 Implementation

The first thing I have done was to create 2 files: index.html and app.js. Before I started to write the JavaScript code I created a basic HTML skeleton with tags like <head>,<body>,<p>, <a> and fields for input. After this I have copied the cdn link of angular, angular-router, jquery and bootstrap libraries at the top of my index.html.

Based on the tutorials I have done before starting my own project I have created first HomeController in app.js file and practiced displaying objects from the database, that I have added manually not through the code, on the front page of my web.

```
  <div ng−repeat="post in posts">
    {{post}}
  </div>
```

Next step was to create a function to add the post into the database. On the index.html I have added an ng-click="addPost()"function, and ng-models, and I have used $scope to allow controller to get access to text boxes from the index.html. At this point I was not sending the information to the database yet only created a way to receive and display text.

From one of the tutorials I have learned about ng-repeat function that I have implemented next for automatic repetition of a piece of code to display all the instances of posts on my website.

Next step was to start implementing services on my back end. Again, based on the tutorials I have implemented a factory of posts instead of exporting the array of posts directly. This way I could further expand services and add new methods in the future to the post array.

Because I had ui-router library included, I could configure in the app.js a config() function to setup a home state. To move from one state to another I have created a second html page and named it 'home'. This way I created a template that will be used to display the home view and referenced it in my JavaScript code. The idea behind this is that whenever ui-router detects a route change, it will place the new template inside the <ui-view> tag located in index.html and initialize the controller specified in the state configuration. Similarly to my home page I have also created a template and controller for displaying a single post and post editing.

Now that I had the basic front-end for my blog coded up, I have started to work on the backend and communication with the database. Using 'express-generator' I have created a new nodeJS project:

```
express −−ejs Blog
cd Blog
```

Ejs flag allow the server to use standard HTML in the templates.

When starting a new project, a generator will include a list of packages that are required in the file called packages.json and store them in node-modules when a command npm install is invoked. My next step was to install a mongoose library that provides schemas and models on top of MongoDB. When I had all the packages in place I have moved my html files to the views directory and changed the extension from html to ejs. Next I moved the app.js to a public folder. From now on I could run my application using node app.js command and view it on the http://localhost:3000.

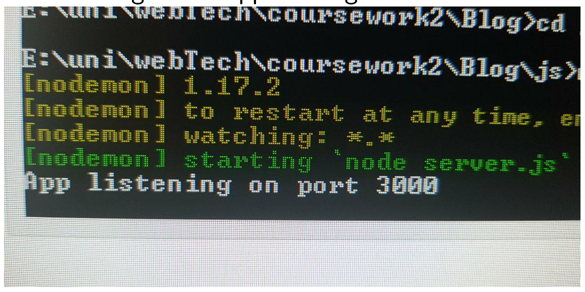


Figure 2: App running on nodeJS

```
E:\uni\webTech\coursework2\Blog>cd

E:\uni\webTech\coursework2\Blog\js>
[nodemon] 1.17.2
[nodemon] to restart at any time, en
[nodemon] watching: *.*
[nodemon] starting `node server.js`
App listening on port 3000
```

To store the data in a MongoDB I included a mongoose.connect with the address copied from mlab. This will open a connection to my myblog2018 database. Mongoose acts as a front end to MongoDB and is object modeling tool designed to work in an asynchronous environment [5].

To use the database effectively I had to create a Post model that will tell the database how and what information to store when saving a post and added this model as a required in my app.js file.

```
var PostSchema = new mongoose.Schema({
 title:{type: String, required: true},
 body: String,
 author: String,
 posted: {type: Date, default: Date.now}
});
```

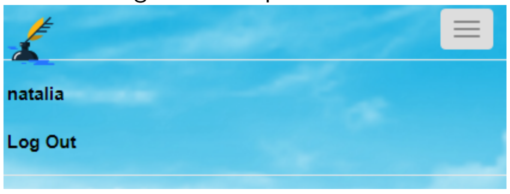
To navigate through the website and perform an action routes are necessary. View all the posts will use GET route and update a post will require a PUT route in place. To create a route I have used the express get(), post() and put() methods, defined the URL for the route and a function that will handle the request. Inside the routes a database is queried and if an error occurs, it is passed to the error handling function. When a route is created two variables are passed to it: req (request) and res (response).The request contains information about the request that is made and the response has information send to the client. A lot of the templates for creating routes, controllers (and more) I have found on the Angular Docs[2], and AngularJS [4] websites.

Now that the basic POST, GET and PUT routes were created I returned to the posts factory and added new functions that will allow retrieving all the posts, a single post, and to create, delete and edit one. When these basic functions were in place I have implemented a user model and authentication based on one of the tutorials [6]. Adding a user let me associate Posts with its author and use ng-hide/ng-show options in the HTML to hide/show forms and divs from users that are not logged in.

After I have finished implementing authentication I have tested registration and login routes using Postman program. The register and login testing result are displayed in the appendices: Appendix A,Appendix B, Appendix C.

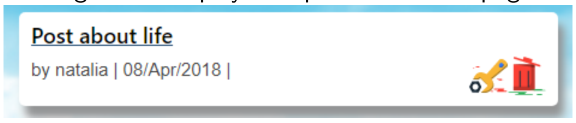
The final part of my implementation was to work on page design. Using bootstrap I could quickly insert a style for my navigation bar that will be responsive and jquery library allow for the button dropdown option when the window is to small too display full bar.

Figure 3: Dropdown menu



I have added a wrapper to display posts and instead of regular buttons for editing and deleting I have used icons.

Figure 4: Display of a post on a main page



More screenshots of the final design of the website are included in appendices starting with Appendix D.

4 Critical evaluation

Comparing to the original idea of my application I managed to implement all of the planned features. The minimum required for the app to be usable - CRUD and a multi-user support. This gives more control on who can add, delete or edit a post. I also implemented some of the elements of responsive design. All the elements scale with the window, including the navigation bar.

- Improvement to the app would be adding a search option. The more posts will be added, the longer it will take to find the one that the user is interested in. The displayed list sorts the recipes in the ascending order by date, so reaching an older post may be a challenge.
- A nice addition to the app would be also an option to display only a certain amount of posts per page and add next and previous buttons to navigate. With the increasing number of posts time to load the front page will increase as well.
- Another improvement could be adding a comment option to allow user give their opinion about the post they just had read.

On the other hand, what I think is really good about my app is that the implemented features are working really well and the overall design is pleasant and clean. I believe that every user will find it easy to use and navigate.

5 Personal evaluation

During the whole process of creating this application I have learned a lot of new things about building a website. It is much more complicated than I initially imagined it would be.

At the beginning I struggled to understand the purpose of the MVC architecture, but after going through a couple of examples I noticed it is a very similar concept to the object oriented programming.

Thanks to the previous coursework, implementing the visual part was a lot easier and faster as I knew what and how to use to make the website look the way I want.

The best part of this exercise for me was creating something very close to what a modern website would look like.

When I compared the implementation and design of this website wit the previous coursework I have noticed how much I have learned and improved my coding since then. My previous website wasn't responsive and could run only from the files located on the disk. This website is far more advanced and structured.

6 Bibliography

References

[1] Acosta, D., *MEAN Stack Tutorial with Angular 2* , 2017 https://www.youtube.com/watch?v=G_xHi0jywmc

[2] Angular.io *Angular Docs* , 2018 https://angular.io/guide

[3] Annunziato, J., *Creating a Simple Blog Using MEAN Stack - webdev fall 2016* , 2016 https://www.youtube.com/watch?v=o_1-Kge54K4

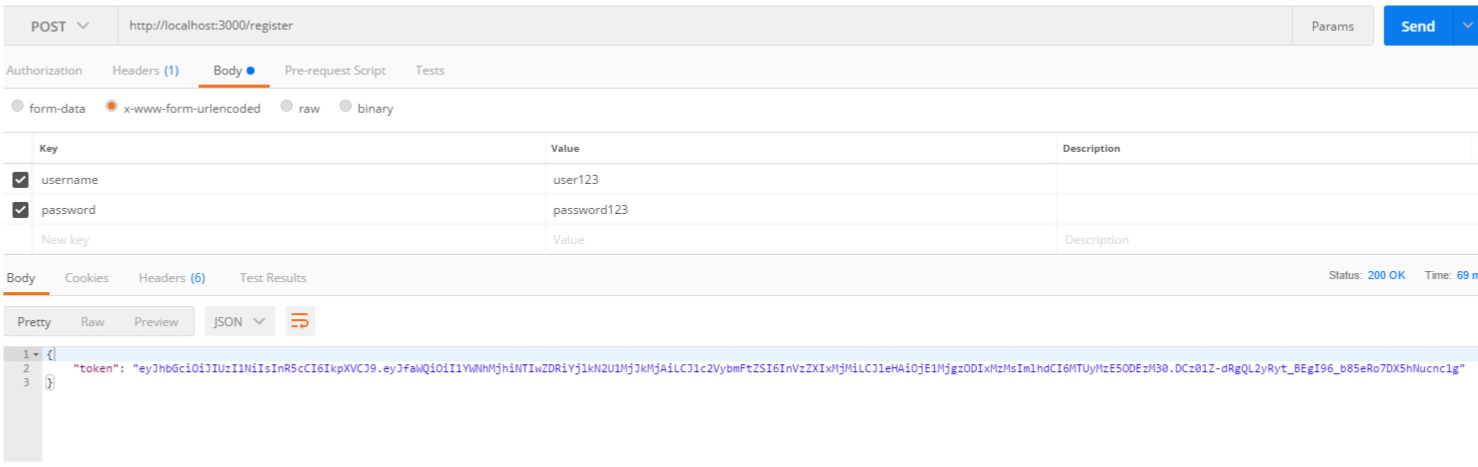
[4] Docs.angularjs.org *AngularJS* , 2018 https://docs.angularjs.org/api

[5] MDN Web Docs *Express Tutorial Part 3: Using a Database (with Mongoose)* , 2018 https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/mongoose

[6] Thinkster.io *Learn to Build Modern Web Apps with MEAN* , 2018 https://thinkster.io/tutorials/mean-stack

Contents

A Registration Test



POST http://localhost:3000/register

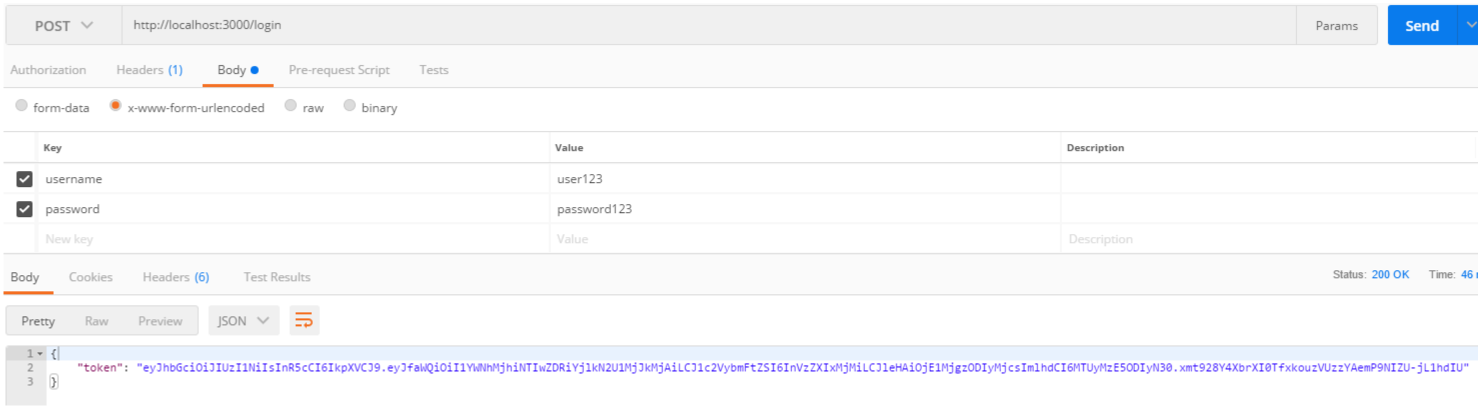
( ) form-data (●) x-www-form-urlencoded ( ) raw ( ) binary

	Key	Value	Description
✓	username	user123	
✓	password	password123	
	New key	Value	Description

Status: 200 OK Time: 69 m

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI1YWNhMjhiNTIwZDRiYjlkN2U1MjJkMjAiLCJ1c2VybmFtZSI6InVzZXIxMjMiLCJleHAiOjE1MjgzODIxMzMsImlhdCI6MTUyMzE5ODEzM30.DCz01Z-dRgQL2yRyt_BEgI96_b85eRo7DX5hNucnc1g"
}
```

B Login Test with correct values



POST http://localhost:3000/login

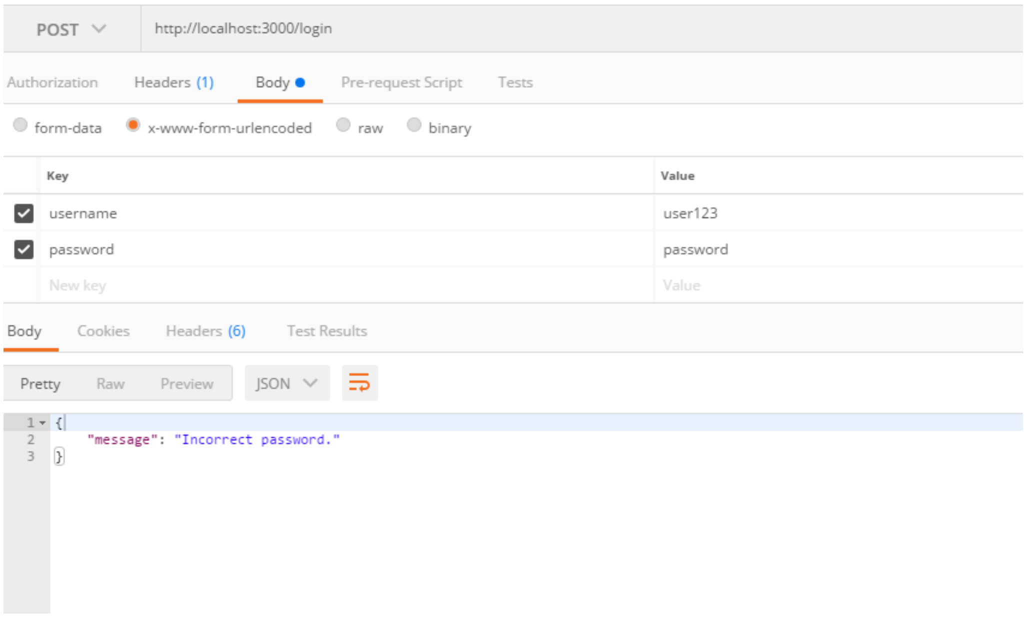
( ) form-data (●) x-www-form-urlencoded ( ) raw ( ) binary

	Key	Value	Description
✓	username	user123	
✓	password	password123	
	New key	Value	Description

Status: 200 OK Time: 46

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI1YWNhMjhiNTIwZDRiYjlkN2U1MjJkMjAiLCJ1c2VybmFtZSI6InVzZXIxMjMiLCJleHAiOjE1MjgzODIyMjcsImlhdCI6MTUyMzE5ODIyN30.xmt928Y4XbrXI0TfxkouzVUzzYAemP9NIZU-jL1hdIU"
}
```

C Login Test with incorrect values



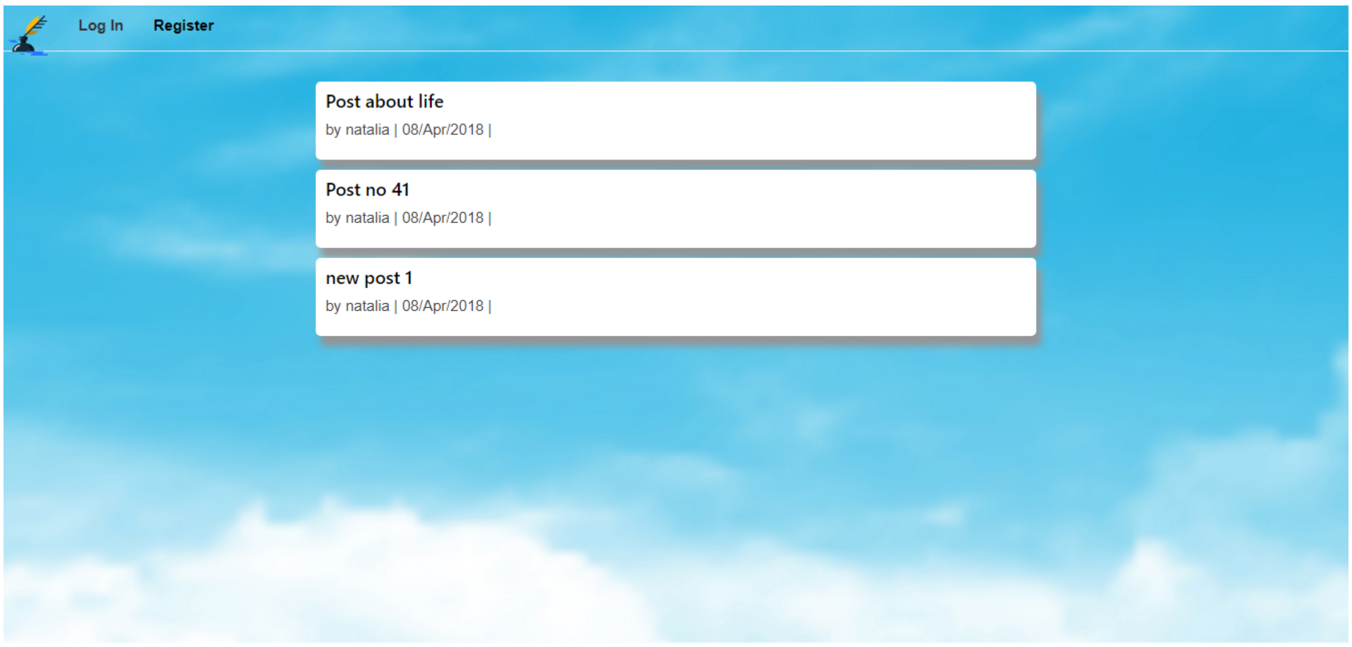
POST http://localhost:3000/login

( ) form-data (●) x-www-form-urlencoded ( ) raw ( ) binary

	Key	Value
✓	username	user123
✓	password	password
	New key	Value

```
{
    "message": "Incorrect password."
}
```

D Home page if not logged in



Post about life

by natalia | 08/Apr/2018 |

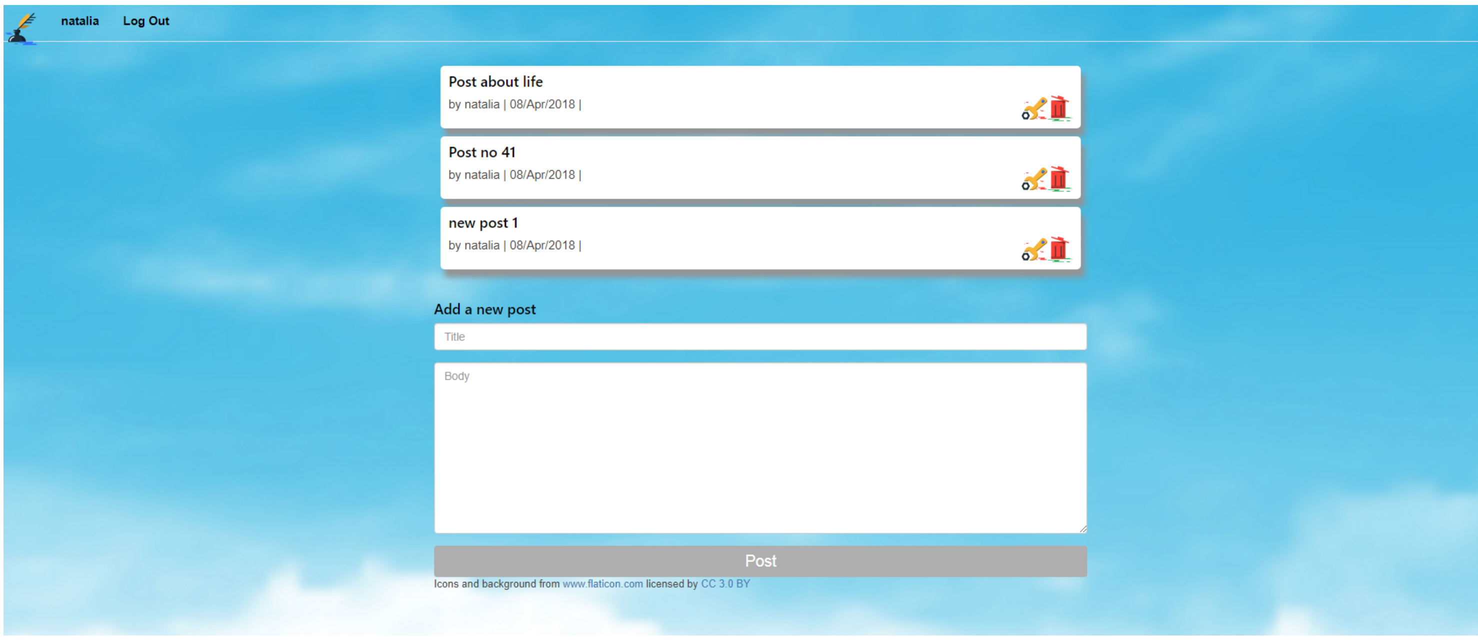
Post no 41

by natalia | 08/Apr/2018 |

new post 1

by natalia | 08/Apr/2018 |

E Home page when logged in



Post about life

by natalia | 08/Apr/2018 |

Post no 41

by natalia | 08/Apr/2018 |

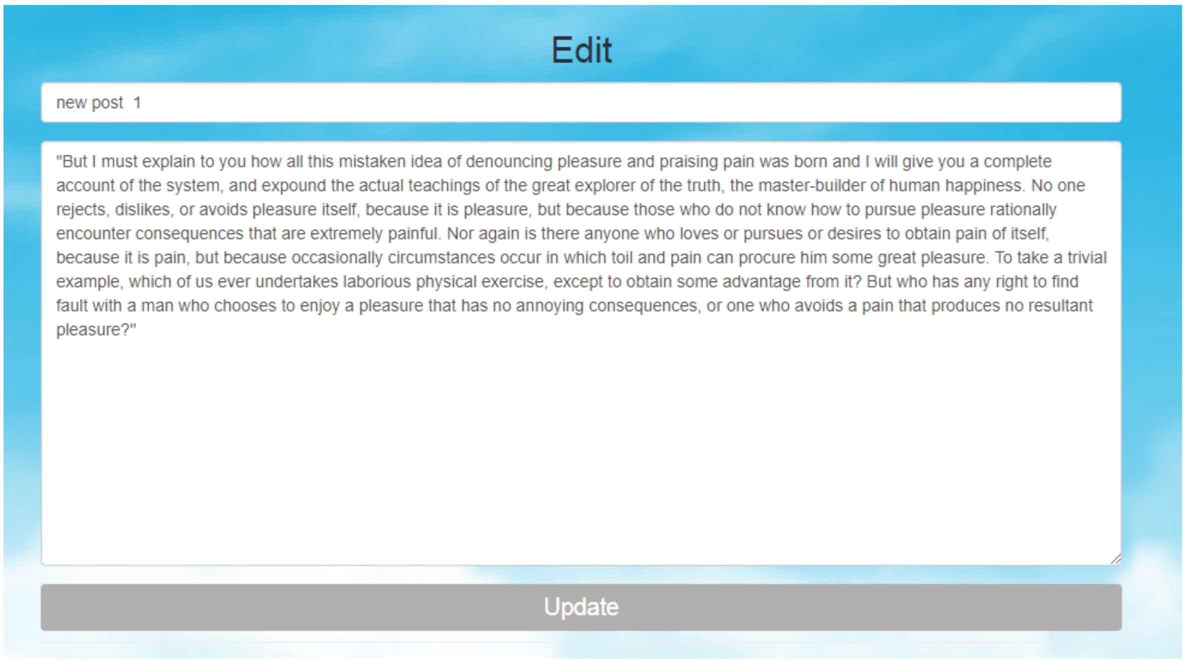
new post 1

by natalia | 08/Apr/2018 |

Add a new post

Icons and background from www.flaticon.com licensed by CC 3.0 BY

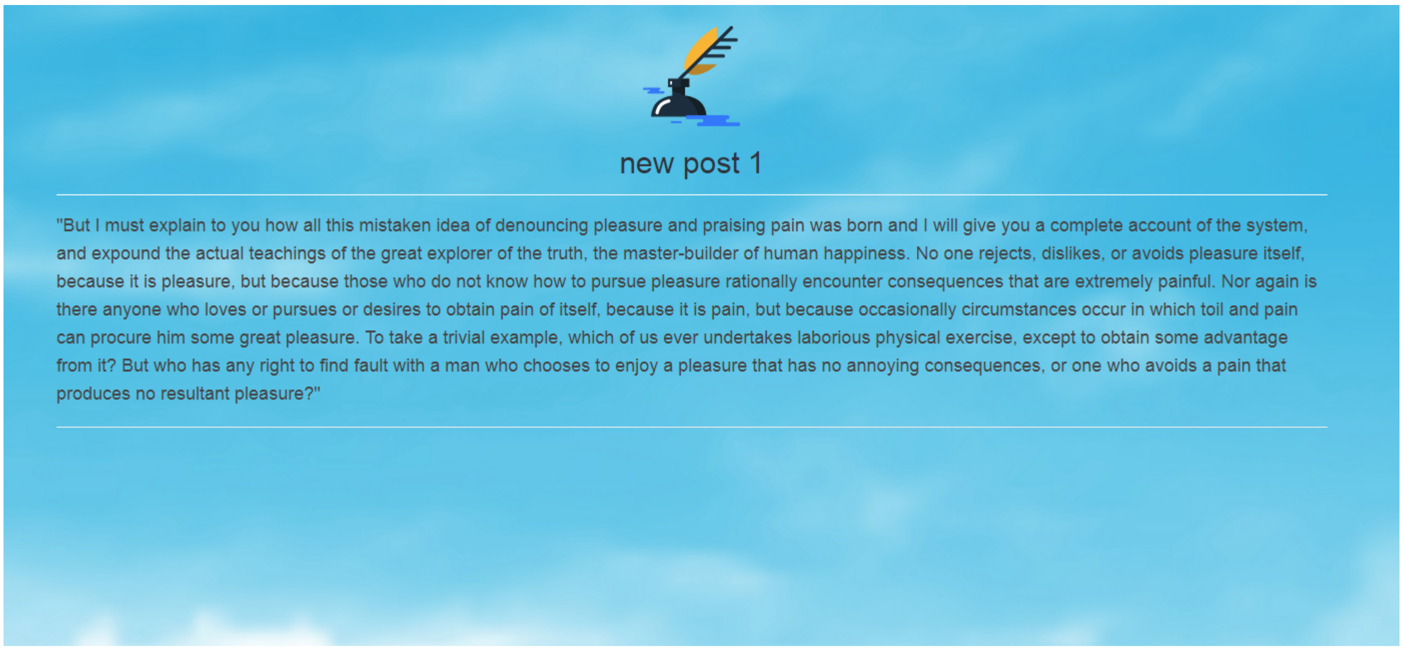
F Edit page



new post 1

"But I must explain to you how all this mistaken idea of denouncing pleasure and praising pain was born and I will give you a complete account of the system, and expound the actual teachings of the great explorer of the truth, the master-builder of human happiness. No one rejects, dislikes, or avoids pleasure itself, because it is pleasure, but because those who do not know how to pursue pleasure rationally encounter consequences that are extremely painful. Nor again is there anyone who loves or pursues or desires to obtain pain of itself, because it is pain, but because occasionally circumstances occur in which toil and pain can procure him some great pleasure. To take a trivial example, which of us ever undertakes laborious physical exercise, except to obtain some advantage from it? But who has any right to find fault with a man who chooses to enjoy a pleasure that has no annoying consequences, or one who avoids a pain that produces no resultant pleasure?"

G Single post page



new post 1

"But I must explain to you how all this mistaken idea of denouncing pleasure and praising pain was born and I will give you a complete account of the system, and expound the actual teachings of the great explorer of the truth, the master-builder of human happiness. No one rejects, dislikes, or avoids pleasure itself, because it is pleasure, but because those who do not know how to pursue pleasure rationally encounter consequences that are extremely painful. Nor again is there anyone who loves or pursues or desires to obtain pain of itself, because it is pain, but because occasionally circumstances occur in which toil and pain can procure him some great pleasure. To take a trivial example, which of us ever undertakes laborious physical exercise, except to obtain some advantage from it? But who has any right to find fault with a man who chooses to enjoy a pleasure that has no annoying consequences, or one who avoids a pain that produces no resultant pleasure?"