1 Introduction

Within today's society people are easily offended. With a world full of offended people, feelings run high and people require a place to post their thoughts, opinions and reasoning that are going on within their day to day life.

A blog is the perfect way for users to post their personal views for the world to see. Blogs allow for a variety of method flexible and defined to the users interests, topics and variety of posts that they may wish to share.

It is important that the user is able to create, edit, update and delete posts that they make. For example, a controversial post on their blog may receive a negative response and the user may wish to either update their post or delete it all together.

This application will use HTML, CSS, NodeJ, EJS and MYSQL to create a functional blog which is accessible by multiple users which could be used in a variety of ways.

A simplistic design will be used to enhance user experience. Clear instructions will be provided on how to use the blog as buttons and hyperlinks are clearly marked on their meaning. For example, "Create Post", "Edit", "Delete" clearly mark out the intended purpose of the function.

2 Software Design

The design of the webpage aims to give the user a good experience. This includes efficient color schemes to make the application feel pleasant, easy to read text and clear instructions of international buttons found within the application.

2.1 Approach

The following languages will be used to code the website alongside their usage;
1. HTML - HTML will be used to provide the structure of the application. This includes setting out headers, footers, containers and dividers where content will be placed.
2. CSS - CSS will be used to design the website. This includes the color scheme which aims to be pleasing, positioning of containers and apply the same style to repeated items throughout the application.
3. BootStrap - Bootstrap is a library which will be used to construct the layout of the website. Boostrap allows for scalability with their responsive grid system and pre-designed items which give a website/application a professional feel with ease.
4. Javascript - Javascript will provide the webpage with a "back to top" functionality which will show when the user scrolls down as blog posts may become lengthy.
5. Embedded Javascript - EJS will be used to read the data from the database. This allows for javascript to be placed within the html of the website and provide a for loop to read each post stored.
6. MySQL - MYSQL will be used to store the posts which are made within the blog.

2.2 Design Requirements

The webpage will be designed and created with the following requirements;
1. Navigation:

- Placed on the top of every page
- Allow the user to access key points within the webpage.
- Provide feedback when the user hovers over the navbar.

2. Database Querying:

- Users must be able to CREATE posts.
- Users must be able to READ posts.
- Users must be able to UPDATE posts.
- Users must be able to DELETE posts.

3. Users

- One more users must be supported within the application.

2.3 Templates

Templates will be used to create a repeating theme throughout the application. This enhances user experience each webpage should contain the same generic style. The following templates will be used;

1. **Header**
   A header template will be used to provide the same header throughout the application. This includes the following;
   - Opening HTML and Body tags which will not be repeated manually on each page.
   - Import CSS, Javascript and Boostrap throughout every webpage.
   - Navigation at the top of every webpage.
   - Same title throughout each page - Set08101 Coursework Submission.
   - Same gold header at the top of each page.
   - Allow for "back-to-top" scrolling feature on every page.
2. **Footer**
   - Closing body and html tags

2.4 Color Scheme

The color scheme of the website was chosen with the aim to avoid strain on the users eyes. Each color complements each other along with the color of the text to be easy to read. By doing this sharp, bold and bright colours are avoided and more "pastel" like colors had been selected as seen below.

2.5 Initial Design

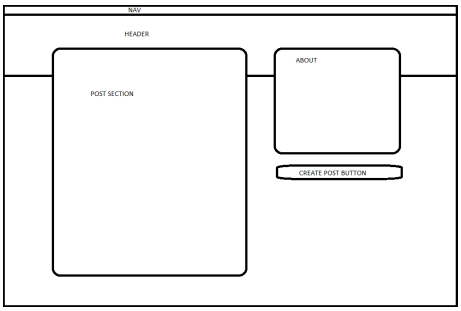

Figure 1: Blog List Sketch

Figure 2: Create/Add Post Sketch

2.6 Back to Top

The design features a back to top button which appears when the user scrolls down. This is beneficial as blogs may become lengthy. This allows for the user to simply click the button to return to the page.

3 Implementation

3.1 Design Implementation

The website was implemented firstly be writing the HTML of the webpage. Boostrap will be used to create a grid like structure to enable containers within the application to be scalable and also fit neatly within the maximum dimensions of the webpage. This allows for runner space at the left and right of the page with 20 percent margin on each side to fit the application in the centre of the page.

3.1.1 Homepage

The homepage allows the central hub for the users interaction of the website. Firstly, the homepage allows the viewing of the blog ordered by newest first. Users can view the post title, content and the author who had written the post.

The homepage also allows for users to create posts which will redirect the user to /blog/add where a post creation form is created. Lastly, users can edit and delete posts from the homepage. The homepage layout can be viewed below.

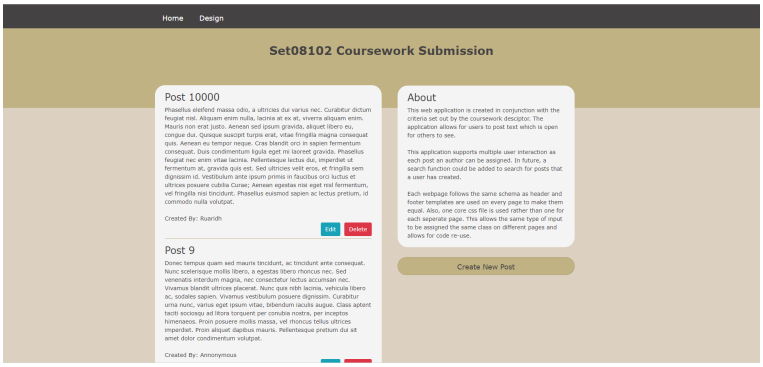

Figure 3: Homepage

3.1.2 Create Post

The create post page allows user to create a post. Within this page users can define a title, content and author of the post that is being created. However, as this blog is openly shared with multiple users some data may be sensitive to the user and therefore by default users can post anonymously if no author is defined.

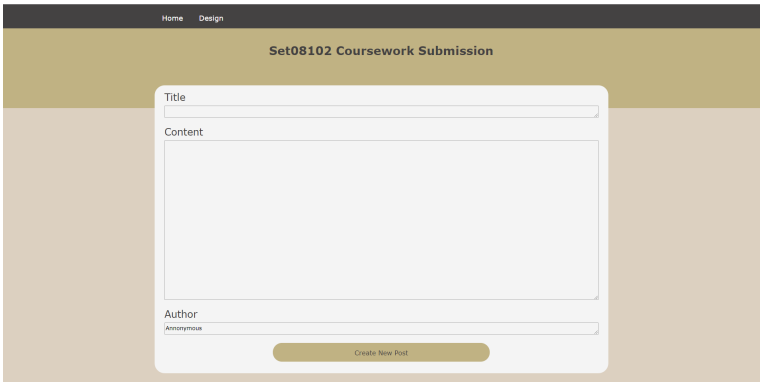

Figure 4: Create Post

3.1.3 Edit Post

The edit post page allows users to edit a post. This could be beneficial to fix spelling/grammar mistakes, update a post throughout the day or add extra information to a post. For example, a user may post an issue they are having and once the issue is fixed put a notice at the top of the post. As seen in the create post section an author can defined. However, within the edit page this cannot be editted. Once someone creates a post, I do not believe it should be able to be changed as the post belongs to that user.

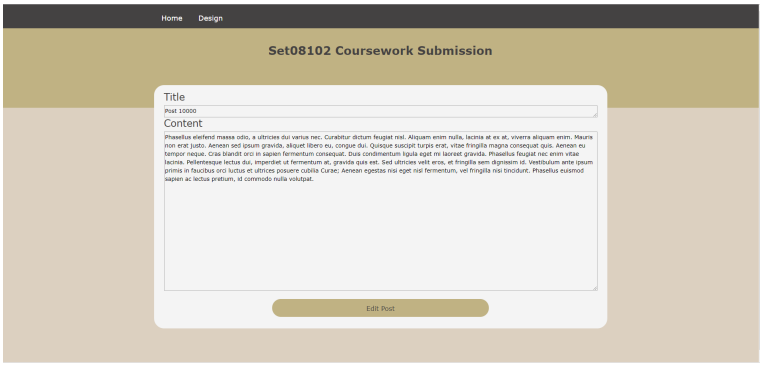

Figure 5: Edit Post

3.2 Database

Posts which the users create will be stored within a locally hosted MYSQL database which will allow queries to be ran to give the desired data output.

Data will be stored within the blog database and within the blog_post table with the following structure;

Column	Datatype	Properties
id	int(11)	auto increment, PK
name	varchar(100)	NOT NULL
content	text	NOT NULL
author	varchar(40)	NOT NULL

Users will run CRUD queries on the database as described within the next section.

3.3 CRUD Implementation

CRUD is a terminology defined for users interaction with data. This describes 4 basic methods users should be able to do with the data stored as described below;

- **C**reate
- **R**ead
- **U**pdate
- **D**elete

nodeJS is used for the backend server setup which will allow the user of the application to be able to carry out 4 key requests to the server - GET, POST, PUT, DELETE. Each of these requests allow for CRUD to be carried out and are detailed in the following sub-sections.

3.3.1 Create

To create a new blog post, a POST request is sent to the server with the data defined by the user. To allow the data to be attached to the POST two things must be setup.

1. Variable A variable is set to the data which is to be stored within the database. The data is escaped to avoid an SQL Injection attack and also trimmed to take out extra spaces. The variable setup can be viewed in Listing 1: Variable Setup

```
var bPost = {
        name: req.sanitize('name').escape().trim(),
        content: req.sanitize('content').escape().trim(),
        author: req.sanitize('author').escape().trim()
    }
```

Listing 1: Variable Setup

2. Query The database must be queried for the data to be inserted into the database. This query will take take data stored within the above variable and insert it into the request (Listing 2: Create Query).

```
req.getConnection(function(error, conn) \{
        conn.query('INSERT INTO blog_post SET ?', bPost,
    \}
```

Listing 2: Create Query

3.3.2 Read

To be able to read the post data which is stored in the database a GET request is made to the server. This request fetches the data from the database and allows it to be printed to the database. A query must be performed to the database for the relevant data to be retrieved (Listing 3: GET Query).

```
1 SELECT * FROM blog_post
2 ORDER BY id DESC;
```

Listing 3: GET Query

Although requesting a query to the database via a GET request this does not allow the data to be printed to the webpage. A for loop using Embedded Javascript (EJS)is used to read through read row of data and print it to page (Listing 4: Read For Loop)

```
1 <\% if (data) \{ \%>
2       <\% data.forEach(function(bPost)\{ \%>
3       <h3><\%= bPost.name \%><br /></h3>
4       <\%= bPost.content \%><br /><br />
5       Created By: <\%= bPost.author \%><br />
```

Listing 4: Read For Loop

3.3.3 Update

Update uses a PUT request to the database with the updated data. Much like Create a similar methods applies with using the bPost variable(Listing 1: Variable Setup). However, the author is excluded as this does not require to be updated. Also, a query must be sent to the database in order for the row to be updated (Listing 5: Update Query)

```
1 req.getConnection(function(error, conn) {
2       conn.query('UPDATE blog_post SET ? WHERE id = ' + req.params.id, bPost,
```

Listing 5: Update Query

3.3.4 Delete

Delete is similar to Create and Update however is more simpilar due to the only parameter required is the id of the post (Listing 6: Delete Variable).

```
1 var del = \{ id: req.params.id \}
```

Listing 6: Delete Variable

The delete sends a DELETE request to the database with what row to delete defined by the id of the post (Listing 7: Delete Query).

```
1 req.getConnection(function(error, conn) \{
2       conn.query('DELETE FROM blog_post WHERE id = ' + req.params.id, del,
```

Listing 7: {Delete Query}

4 Critical Evaluation

4.1 Requirements

The requirements of this webpage set out in Section 2.2 are displayed below with whether they have been completed or not.

4.1.1 Navigation

- ☑ Placed on the top of every webpage.
- ☑ Allow users to access key points within the webpage.
- ☑ Provide feedback when hovering.

4.1.2 Database Querying

- ☑ Users must be able to CREATE posts.
- ☑ Users must be able to READ posts.
- ☑ Users must be able to UPDATE posts.
- ☑ Users must be able to DELETE posts.

4.1.3 Users

- ☑ One of more users must be supported within the application (Section 4.2)

4.2 Users

As mentioned in section 4.1.3. users have been completed as desired by the requirements. However, the implementation of users is not entirely 100% legitimate.

Multiple users are supported due to the fact an author can be assigned to each individual post. This allows for multiple users to share their posts with each other and give each post a persona as a user.

Security is an issue with this implementation as there is no password protected information. Every user can access everyones data and read everyones posts, some people may wish their post to be private. Every user can edit and delete every post. Therefore, someone could write a heart wrenching post for someone else just to delete it. Improvements of this will be discussed in section 4.3. Improvements.

4.3 Improvements

As this application is supposed to be a blog a lot of features can be added to enhance the functionality of the blog. Listed below in each subsection is potential improvements that could be made to the blog.

4.3.1 Users

As explained in section 4.2. Users are a big improvement which could be made to the blog. A login system would enhance security and protect peoples posts and data. Within this login system an administrator account would be beneficial to maintain the user accounts.

4.3.2 Individual Feeds

Within the current design all posts are viewed together. However, with the implementation of user accounts individual blogs can be created which will view only posts belonging to that user. However, a full listing of blogs could also be viewed by the user. A psuedocode of this can be viewed in Listing 8: User Defined Posts.

```
1  var uID= {
2          userID: req.sanitize('userID').escape().trim(),
3      }
4
5  req.getConnection(function(error, conn) \{
6      conn.query(SELECT FROM users WHERE userID = ' + req.params.id, uID,
```

Listing 8: User Defined Posts

4.3.3 User Defined CSS

Within modern day blogs users can make their blog layout personal to them. With the implentation of Users Accounts the blog could have been more custom with user defined css which will allow the user to change the background colors, fonts and perhaps upload images for backgrounds.

4.3.4 Editable Posts

Currently posts are crammed into the one block text. An improvement of this would allow posts to be allow line breaks, bold, italic, font changes and file uploads for the post.

5 Person Evaluation

Personally I feel this coursework was a challenge. However, once I understood the basics it became easier to initiate the GET, POST, PUT and DELETE. The blog could have had much more functionality but as I struggled to implement user accounts a lot of my ideas could not be achieved.