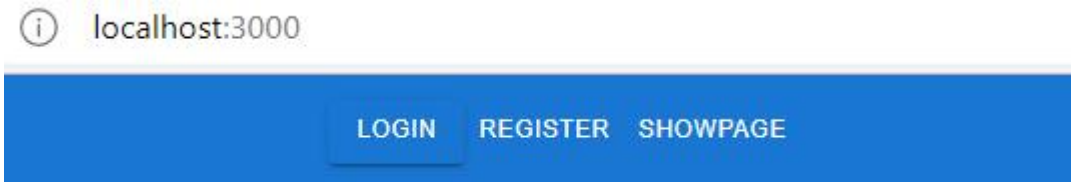
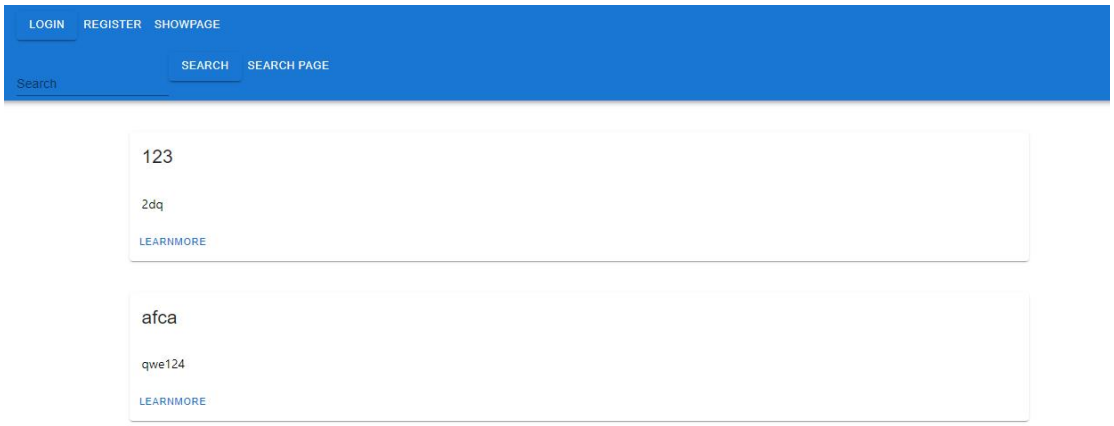
Features I have implemented and the number of points I'm aiming at:

Feature	**Max points**
1.Basic features (as stated in the previous chapter) with well written documentation	25
2.Users can edit their own comments/posts	4
3.Utilization of a frontside framework, --- React	5
4.Use highlight library for the code snippets ---'react-syntax-highlighter'	2
5.Use of a pager when there is more than 10 posts available	2
6.Admin account with rights to edit all the post and comments and delete content (if a post is removed, all its comments should be removed too)	3
7.Test software for accessibility	3
8.Provide a search that can filter out only those messages that have the searched keyword(searched keyword in description part)	2
9.Last edited timestamp is stored and shown with posts/comments	2
10.Create (unit) tests and automate some testing for example with https://www.cypress.io/ (at least 10 cases have to be implemented)	5
<Your own feature:Detailed descriptions are below>	53

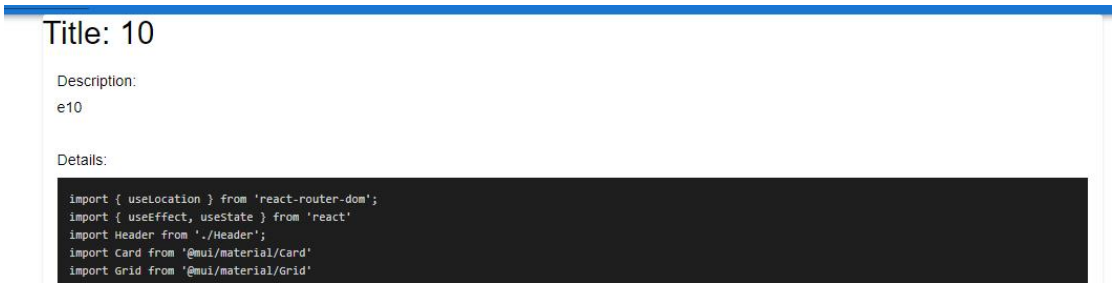
1,Basic features (as stated in the previous chapter) with well written documentation:
I implemented the backend using Node.js and Express, and the database's use MongoDB to store the data
Using Materialize, the entire application is responsive and works on both mobile devices and desktop browsers.
Users can find the registration and login options in the navigation bar when they enter the site



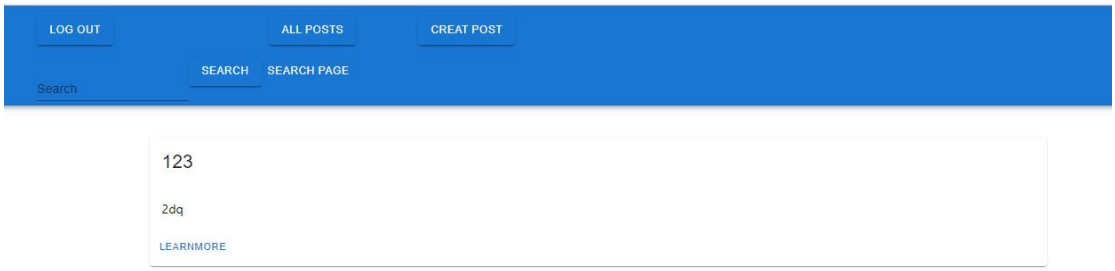
By clicking the button they can go to the registration screen or the login screen.
I use JWT to authorize the logged in users, the backend "/api/create/, /api/edit/" are required for the user to use after logging in. The rest of the paths, such as "/api/get/", can be used without logging in.
Before logging in, users can view all posts on the home page, they can see the title and description of the code, and click LEARNMORE to view the code and all comments of the post.

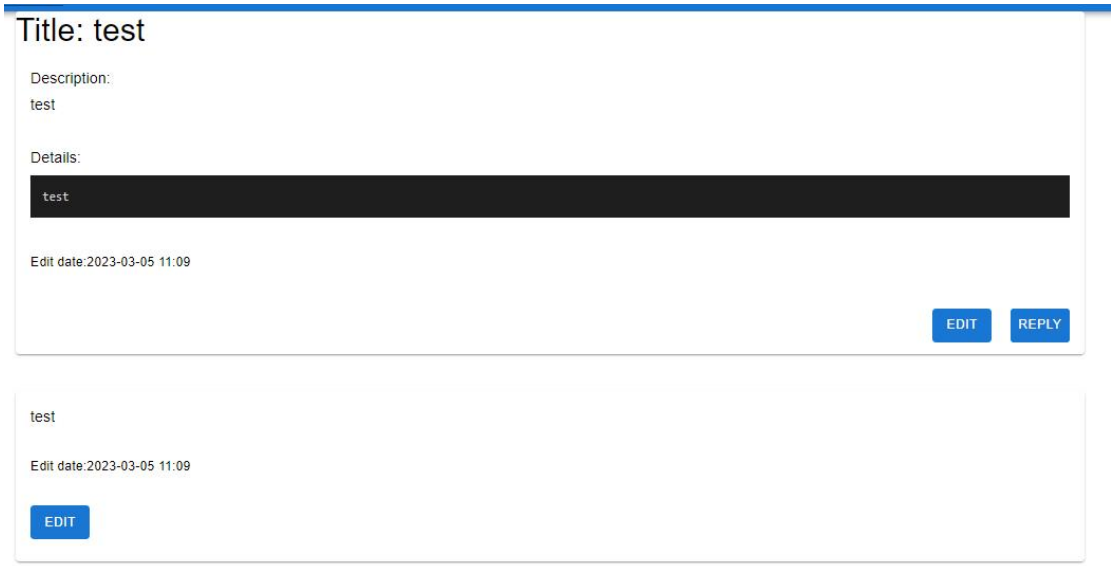


This one is after entering LEARNMORE.



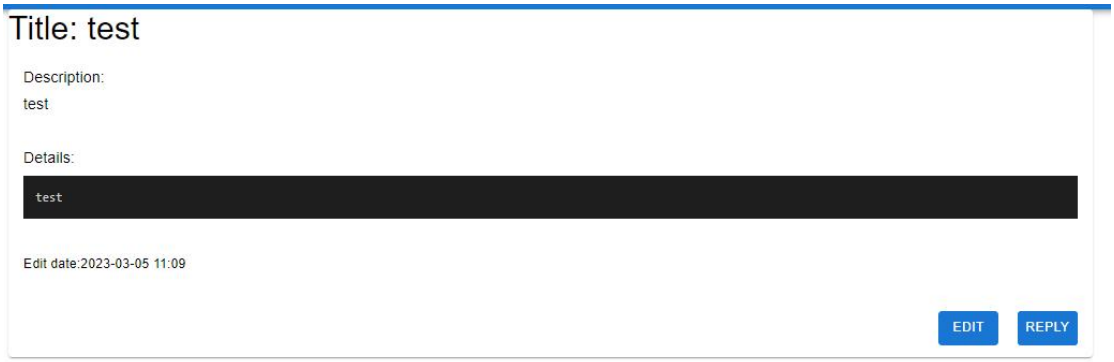
After users register and login, there are options of LOG OUT and CREATE post in the navigation bar, users can post by clicking the button. Click LEARNMORE to enter the post and you can post comments.

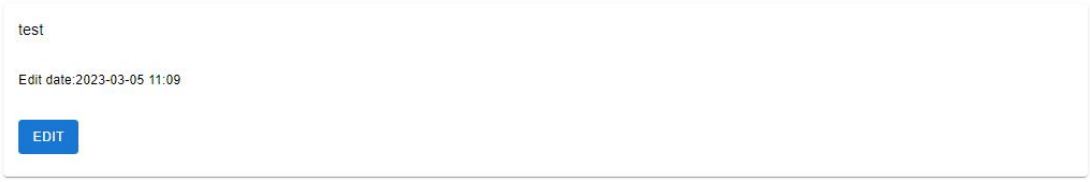




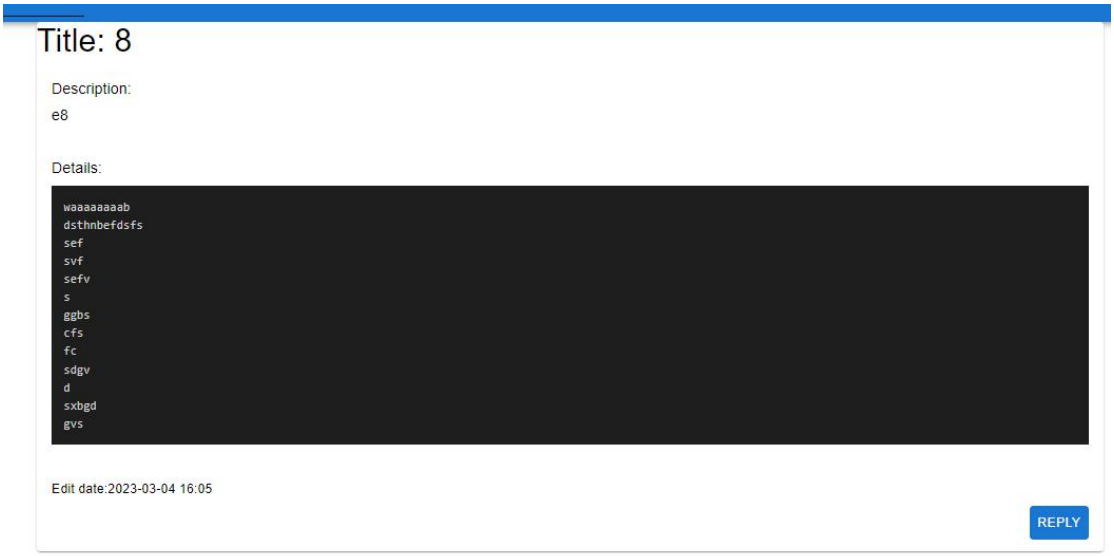
2. Users can edit their own comments/posts

If the post and comment are created by the current user, after entering the post, the EDIT button will appear.





Otherwise there is only the REPLY button:



3 .Utilization of a frontside framework, --- React. Use Materialize, complete with responsive design.

Images are just part of the code

```
return (
  <AppBar position="static">
    <Container maxWidth="xl">
      <Toolbar>
        <Grid container  >
          <Grid item ml={0} mr={20}>
            <Button variant="contained" component="label"
                    onClick={()=>{
                        logoutclick()
                        }}>
                Log Out
            </Button>
```

4. Use highlight library for the code snippets --- 'react-syntax-highlighter'

Code：

```
import { Prism as SyntaxHighlighter } from 'react-syntax-highlighter';
import { vscDarkPlus } from 'react-syntax-highlighter/dist/esm/styles/prism';
```

```
<SyntaxHighlighter language={post[0].language} style={vscDarkPlus}>
{post[0].details}
</SyntaxHighlighter>
```

Example：

Title: 10

Description:

e10

Details:

```
import { useLocation } from 'react-router-dom';
import { useEffect, useState } from 'react'
import Header from './Header';
import Card from '@mui/material/Card'
import Grid from '@mui/material/Grid'

import Container from '@mui/material/Container'
import CardContent from '@mui/material/CardContent'
import Typography from '@mui/material/Typography'
import { Prism as SyntaxHighlighter } from 'react-syntax-highlighter';
import { vscDarkPlus } from 'react-syntax-highlighter/dist/esm/styles/prism';

export default function ALearnmore() {



    const { search } = useLocation();
    const queryParams = new URLSearchParams(search);
    const postId = queryParams.get('postId');
    const arr = postId.split(",");
    const pid =arr[0];
    const aid =arr[1];
    console.log(pid,aid)

    const [post, setPost] = useState([]);
    useEffect(() => {
      fetch('http://localhost:1234/api/get/post/',{
        method: 'POST',
        headers: {
          'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            pid: pid
        })
      })
        .then(res => res.json())
        .then(data => setPost(data))
    }, [pid])
```

5. Use of a pager when there is more than 10 posts available

After the number of posts exceeds 10, a pager will be available for page-flipping operations

Code：

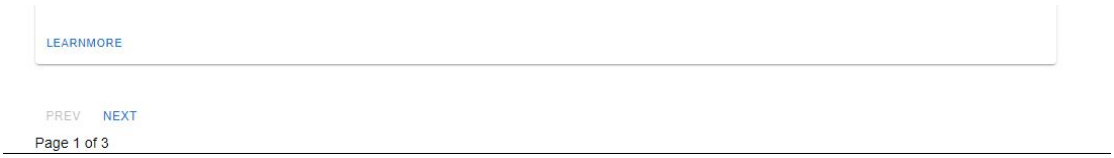
```
const handlePrevPage = () => {
  setCurrentPage(prevPage => prevPage - 1);
};

const handleNextPage = () => {
  setCurrentPage(prevPage => prevPage + 1);
};
```

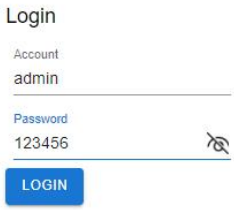

```
const handlePrevPage = () => {
  setCurrentPage(prevPage => prevPage - 1);
};

const handleNextPage = () => {
  setCurrentPage(prevPage => prevPage + 1);
};
```

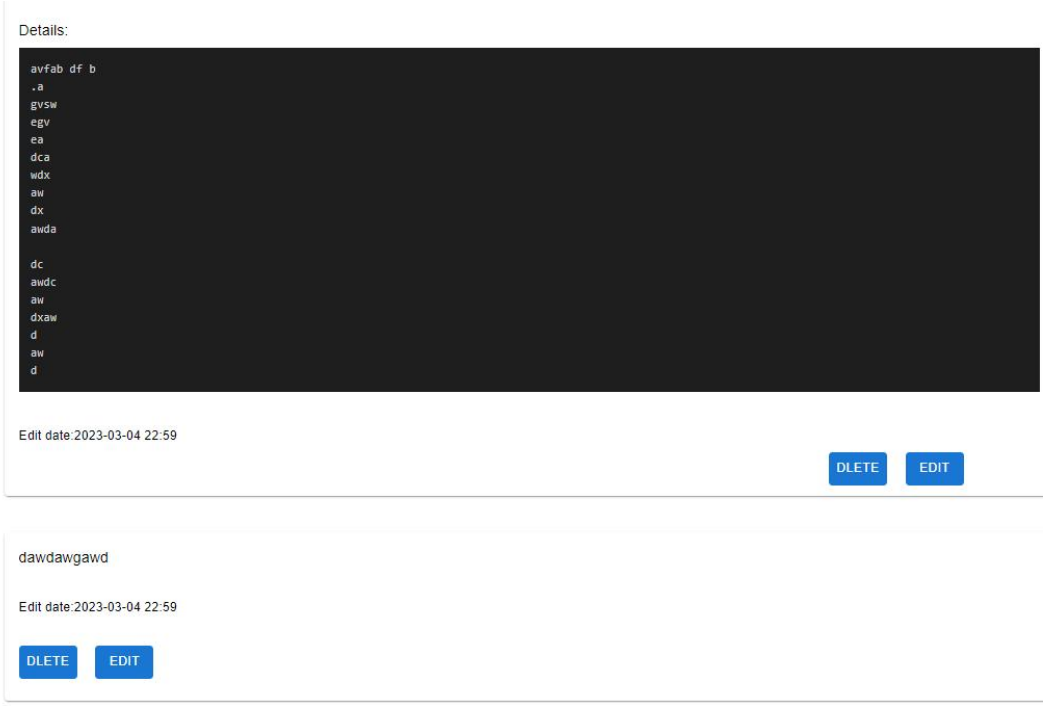

Example：



6. Admin account with rights to edit all the post and comments and delete content (if a post is removed, all its comments should be removed too)
Users in the login screen enter the account input: admin, password input:123456 will enter the administrator interface.



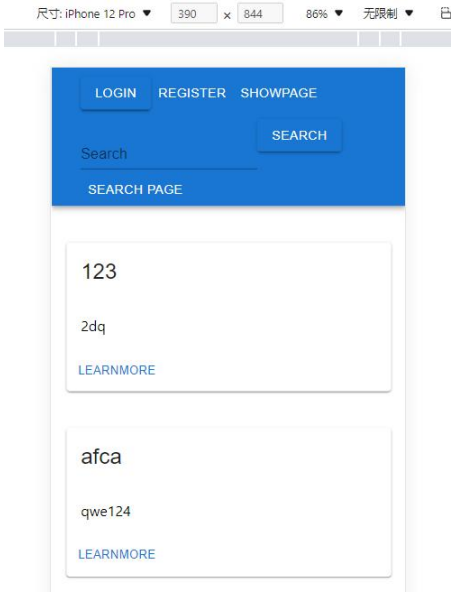
After entering LEARNMOR, the administrator can edit or delete posts or comments.

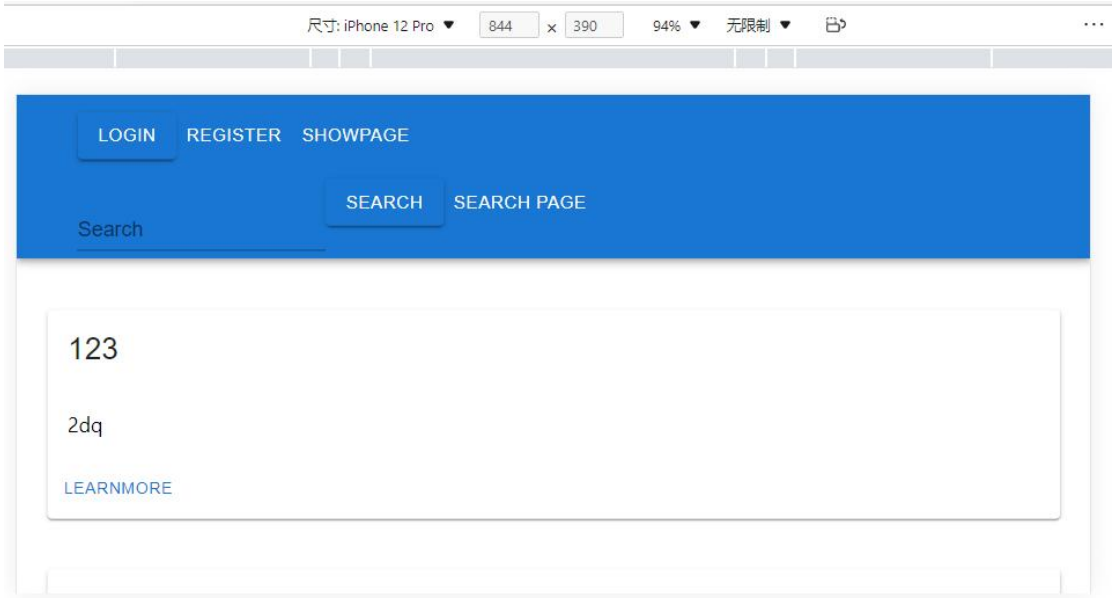


7. Test software for accessibility

I tested it on edge and the software is responsive, but requires a keyboard for input operations.

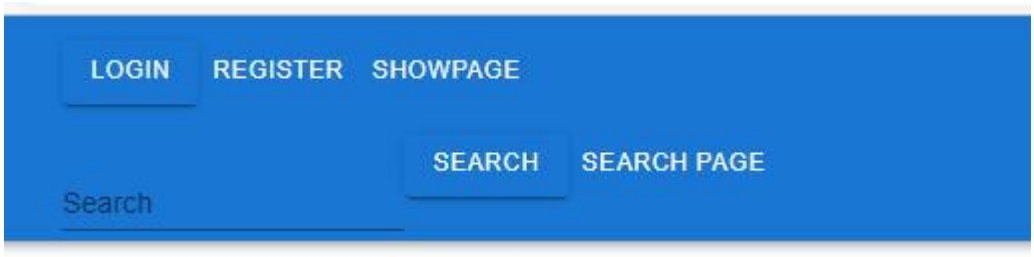
The screen reader works with the application.

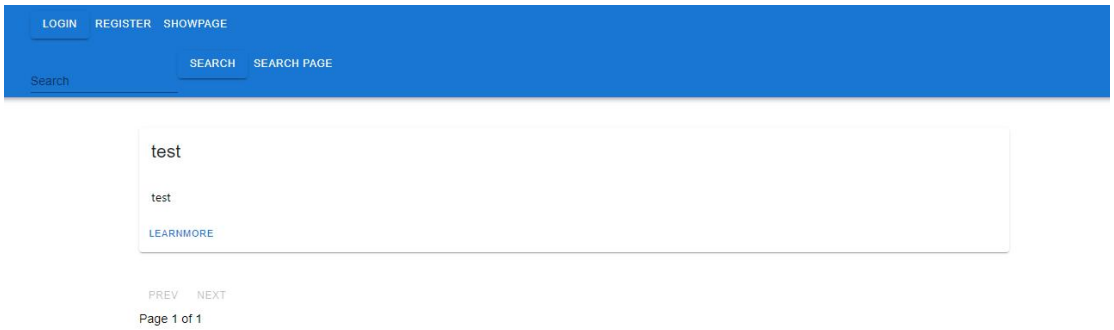




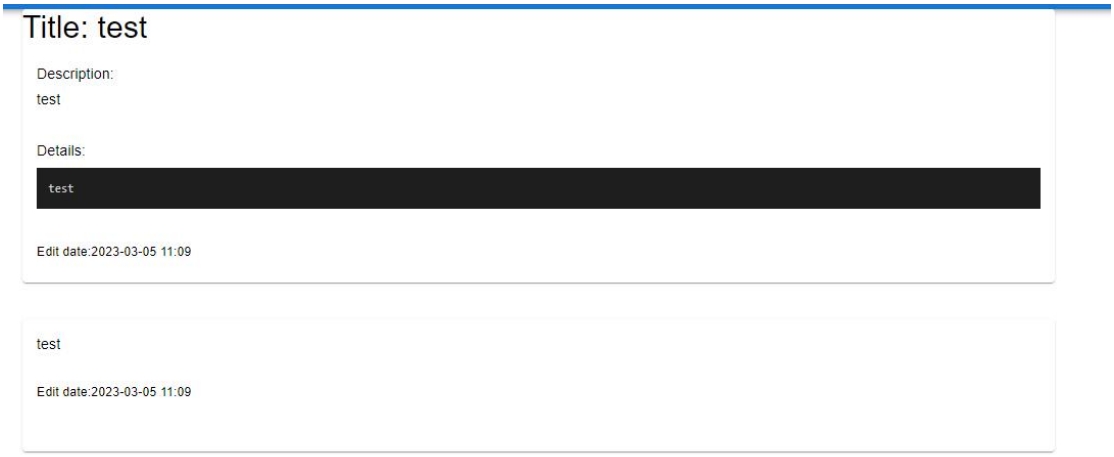
8. Provide a search that can filter out only those messages that have the searched keyword(searched keyword in description part)

The search is present in the navigation bar of any interface , after the user enters the keywords, the application searches the description of each post, and if relevant words exist, they are displayed after clicking the SEARCH button.





9. Last edited timestamp is stored and shown with posts/comments



10. Create (unit) tests and automate some testing for example with https://www.cypress.io/ (at least 10 cases have to be implemented)

As shown in the picture, I have created test cases in Automate_testing folder, there are 15 automation test cases in total. Here is the test video, click the link to open it. Because the properties of some buttons will be changed when the page is refreshed, I changed some code in the background.

Here is the demo video:

auto_test.mp4 (sharepoint.com)