# **REACT – REDUX ASSIGNMENT**

# **Solution Documentation**

Following are the implementation and assumption note for the assignment:

## Requirement:

1. Sign in and sign up functionality work smoothly with the API provided.

User Authentication

Signup

Login

NOTE: Use local Storage to store the Authentication token.

## Implementation:

1. Sign in and sign up functionality working as expected with User Authentication, Signup and Login.
2. If a user signs up in the application, it does not require user to login it takes user to articles page.
3. User can sign out from the application from user settings screen (click on username in the top bar)
4. User can sign in again with the email and password used at the time of sign up process.
5. Application stores user token in the local storage and token expiry is maintained in days. Hence, the user won’t be logged off from the application; unless the user manually signs out from the user settings screen.
6. Front end validations for Sign-In and Sign-Up are implemented through error message sent by the backend server, in case any request fails.
7. Application returns custom message on 401 and 403 status code from server.

## Requirement:

1. Articles: -

Seeing the feed of all articles (route: /articles )

Open and read an article (route:

/articles/{article-name} )

Write a new article

Edit own older articles

Delete own articles

## Implementation:

1. Article functionalities working as expected.
2. Anonymous user can view all the articles; read individual articles and corresponding comments against each article.
3. Signed in user can create a new article, edit an article and delete an article.
4. It will load all the articles via launching the application root URL which is given below.
5. Creating new article makes cache invalid and application re-fetch articles for the first page.

## Requirement:

1. Comments:-

See comments under articles

Write your own comment under an article

Delete own comments

Edit own comments (API Support not available)

## Implementation:

1. User can view comments under an article.
2. Signed in user can post comments on any article.
3. Signed in user can delete it own comment for an article where he/she has previously commented on.
4. Application cache articles per page wise so that it can avoid server request if it is already cached.
5. When user clicks on read more button application checks into the last articles page get the article data so that it can avoid server request.
6. Deleting own article makes cache invalid and application re-fetch articles for the first page.
7. **Note**: User can not edit comment as API support for the same is not available.

## **Application code repository**

<https://github.com/pervezalam777/knowledge-post>

## **Application live link**

<https://knowledgepost-e9776.web.app/>

Note: Deployed using Firebase

## **Good to have features (Pending due to time constraint)**

1. Unit testing (Ideally application should be developed in TDD manner)
2. CI/CD as GitHub and Travis CI can be use for CI/CD deployment
3. E2E testing can be implemented using jasmine and protract
4. Code coverage
5. Loading/processing indications while waiting for server response.
6. Client-side form validation
7. More than 500 articles there in the database so caching implementation in a way so that it only cache last 5 pages data. So that application state does not grow too much which can cause redux reducer processing time.
8. Pagination implementation can be better in terms of UI.