Micro blogging project

Summary

Create a fully functional micro blogging web app (like twitter) in 5 stages

Technical Notes

1. The basic design is provided through Figma. For milestones 1-5 please follow the design provided (making it responsive is an important bonus). For the other milestones you are requested to create the additional features with your own styling.
2. Make meaningful commits on git (they should be small and with feature/bug/milestone specification) and push them to the repo.
3. You will get code reviews on your GitHub repo. Make sure you read this feedback and implement the needed changes/fixing.
4. Keep an organized folder structure - components in “components” folder, pages in another folder and reusable code in lib folder.

Milestone 1 - Mock up

Figma design (first line of screens):
https://www.figma.com/file/KQsukwnsiYo01g7Id1m11Q/ITC-Micro-blogging-Project?node-id=0%3A1

Features:

Main screen with two parts: create tweet (component), and tweets list (component)
The tweet creation should be blocked if there are more than 140 chars (the button should be disabled)
The tweets should be saved locally, so if I refresh the page they won’t be deleted

The tweet list should be sorted in descending order, the latest tweet should appear first (the order should remain after refreshing the page)
Each tweet should include Username. The username should be saved hard-coded for now (so you will be able to add it to each tweet you create)

Milestone 2 - Server connection

Features:

Sign up to https://mockapi.io/ (no need to pay anything, use the free tier). This is a website that creates fully working restAPI for you. You are going to use it in this project in order to save the project's data. Make sure you read the Documentation page before you start working. This is an important guideline for using this service and any other online service that is created and intended for developers.
In MockAPi, create a project & first resource (tweets). Instead of storing your app's data locally, create a connection to your new API. Hint – you should use the GET and POST http methods only. Load the list of tweets from the server, and when the end user creates new tweet – save it in the server.
Show loading indicator and prevent from adding a new tweet while adding request is in the background.
Do not forget to remove the code from the first milestone that saves the data locally, you don't want to have them both.
Display server errors to the user if the tweet is not added.

Milestone 3 - Profile page & Routing

Features:

Add another page that presents the current user username (which should be hard coded until now). This page should have a form which enables to change the username.
You should save the new username locally whenever changed and send it to the server when creating a new tweet.
Add a navbar to the top of the screen that keeps its position no matter which page you are at, with "Home" and "Profile" links.

The design is in the second line of screen in figma:
https://www.figma.com/file/KQsukwnsiYo01g7Id1m11Q/ITC-Micro-blogging-Project?node-id=0%3A1

Milestone 4 - Context & Auto update

Features:

Instead of using local state (normal react's component state [useState]) and props for storing and managing the list of tweets, use internal react context.
When creating new tweet, do not refresh the list from the server. Instead, add the tweet to the existing local list.
In the navbar, add the amount of tweets that have been published so far.
In addition, set an interval that gets the most updated tweets list from the server - in case someone else added a tweet and to keep the list updated. Try to figure out (do a little research and thinking) what is the optimal interval (in millisecond) for this kind of task.

Milestone 5 - Infinite scrolling

Implement infinite scrolling in the tweets page - at the beginning get 10 tweets, and when the user reaches the end of the screen load the next 10 tweets, etc. (hint: look for mockAPI pagination option in the documentation.)

Milestone 6 – Authentication

In this milestone you will have your first experience with adding authentication to your app. It will be simpler than in real apps and without advanced features that we will learn in the future. You can read more about authentication here.

Features:

From now on, only logged in users can see tweets and send tweets, which means you have to implement Login and Sign Up pages. If the user is not logged in, you should prevent the routing to the tweets and user pages and redirect the end user to the login page.
For storing the list of usernames and passwords, create another resource in your mockAPI project. This resource should hold the user id, username, and password.

Since this is your first tryout with authentication, there is no need to hash (encrypt) the password and we don't care about safety/security of the password.
After the end user has registered to the app, he should be able to login and see the tweets & user pages.
When a user is logged in, store his id in local storage. This will be the indication for the app that the user is logged in.
At the right side on the navbar, add a logout button. Whenever the user is logging out, remove his id from the local storage.
Update the profile page – the end user should be able to update his username and password and information should come from the server and should also update the server.
Change the navbar so that it responds based on whether the user is signed in or not. When the user is not signed in, the navbar links should display “Signup” and “Login” links.
Bonus feature – implement automatic login after signup, which means that after a user is finished with the signup process, he will also be logged in and the UI will show him the tweets page.
Implement the authentication feature using context and custom hook.

Milestone 6 - Your tweets

Instead of saving the userName on every tweet, save a reference to the user by the user id.
Implement a feature that allows the end user to display only his own tweets
The feature should have at least the following:

- A clickable element that changes css class based on whether selected or not
  The clickable element should change text based on whether it is selected; when selected, it will show the text "All Tweets", meaning that if you click on it, it will show all tweets; when not selected, it will show the text "My Tweets", meaning that if you click on it, it will show just your tweets.

Upon clicking the clickable element, the page should show the corresponding tweets.
Add one more css class based change to the page upon this event happening; for instance, maybe the page background has a different color when viewing your own tweets when compared to viewing all tweets

Milestone 7 – User's profile image

Add profile picture upload for every user inside the profile page. The image file should be stored online – you can use https://cloudinary.com/ for storing the profile

images or any other service. The image **URL** should be stored with the rest of the user's information inside your backend service - the mockAPI project.
After a new profile picture is uploaded, display it on the profile page and also inside every tweet of this user. Make sure you are limiting the dimensions of the image in the UI so it won't break your design.
Limit the width & the height of images to max of 500*500 pixels (you can do so with clodinary api, search inside the documentation)

Milestone 9 - Like button part 1

Add a like button to every tweet
When a user clicks the like button, the button should change its appearance to indicate to the user that they “liked” that tweet; if a user has not clicked on the button, the button should indicate that the user has not “liked” it
After a user “likes” a tweet, they should be able to “unlike” it
Apply your own styling and text to this button
Optional: Apply an animation that happens when the user clicks the like button

Milestone 10 - Like button part 2

Use your backend API to keep track of which users have “liked” which tweets
When saving “likes” in the backend, be sure to associate the like with the liking user’s id instead of their userName or other identification
Add a "lied tweets" button to the Navbar which will be visible only to signed-in users.
Clicking the button should change the list of tweets to show only the tweets that this user has liked
When the user clicks the button again, the list of tweets should display all tweets
Apply your own styling and text to this button

Milestone 11 - Other users profiles

In the display for each tweet, turn the userName into a link
When a user clicks the userName link on a tweet, your application should take the user to a page that displays the profile for the user who wrote the tweet
The design for this page should be similar in design for the page you made for the User Page, but without an option to edit the user details.

Add a “home” or “back” button to your Navbar so that the user can easily return to the list of tweets

Milestone 13 – Search in tweets (optional)

When the user is signed in, the Navbar should have a search bar.
The search bar should allow users to search the tweets and user; include a button or some way for the user to toggle between searching for tweets and for users
Searching using the navbar should result in the list of tweets displaying only the tweets that match the search criteria
Look inside the mockAPI documentation for how to perform a search inside the tweets list on the server.

Milestone 14 – Comments for tweets - (optional)

Think about what details this feature would need; visit social media platforms that you like and see how they implemented their comments feature.
Implement a way for users to comments after tweets.