

**Micro blogging project**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Summary**

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

**Technical Notes**

1. The design is provided through figma, please follow the design provided (making it responsive is a bonus)
2. Make meaningful commits (they should be small and with feature/bug specification), and push them to the repo.
3. You will get code reviews on your github repo.
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, and tweets list
* Create tweet should block the tweet creation if there are more than 140 chars (need to make the button 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)
* 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:

* Move the local app data (for keeping the data between each refresh), to a server: <https://micro-blogging-dot-full-stack-course-services.ew.r.appspot.com/>
* The server has one resource exposed: “/tweet”, make requests to presenting the list of tweets, and to create a new tweet
* The tweet object is as follows: { content: string, userName: string, date: string (ISO date) }
* Save the tweet to the server on tweet post, and show the list of tweets from the server
* Show loading indicator, and prevent from adding a new tweet when 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 - User page**

Features:

* Add another page that presents the current user username (which should be hard coded until now), and has a form 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**

Features:

* Instead of using state and props, use context for the tweets list and creating new tweet
* When creating new tweet, do not refresh the list, but add the tweet to the existing local list
* Instead set an interval that gets updates from the server of tweets, in case someone else added a tweet (to keep the list updated)

**Milestone 5 - Deployment**

Features:

* Add deployment to firebase so your app will be available from a remote server
* Create a new firebase project (no need to add credit card)
* Follow this guide: <https://medium.com/swlh/how-to-deploy-a-react-app-with-firebase-hosting-98063c5bf425>
* If you have trouble deploying, look for other solutions online for how to deploy a react app to firebase.

**Milestone 6 - Firebase**

* Replace the server connection with firebase firestore
* Send data and get data from firestore (you can use the account you created for deployment)
* Only logged in users can see tweets and send tweets (see “firestore rules”) - implement a Login and Sign Up page with firebase auth, if the user is not logged in - prevent routing to the tweets pages, and redirect the user to the login page. You can implement your own view and design. Implement a login and signup both with google and with a custom email and password.
* Instead of saving the userName on every tweet, save a reference to the user by the user id.
* Add profile picture upload for every user (hint: use firebase cloud storage)
* The data needs to update live when there are new tweets (without intervals)
* Implement infinite scrolling - 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 firestore pagination.)
* No need for a custom backend! all of your code should be in your react project.

**Milestone 7 - Your tweets**

* Implement a feature that allows the user to display only their 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; hint: use state/props and life cycle methods to control which tweets are displayed
* 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 8 - Navbar**

* Change your Navbar so that it responds based on whether the user is signed in
* When the user is not signed in, the Navbar links should display “Signup” and “Login” links.
* When the user is signed in, the Navbar should display “Logout” and also 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

**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
* Apply an animation that happens when the user clicks the like button

**Milestone 10 - Like button part 2**

* Use firebase to keep track of which users have “liked” which tweets
* When saving “likes” in firebase, be sure to associate the like with the liking user’s id instead of their userName or other identification
* Add a button to the Navbar so that when the user is signed in for 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
* Add a “home” or “back” button to your Navbar so that the user can easily return to the list of tweets

**Milestone 12 - More firebase**

* A user should be able to edit their own profile
* When a user is logged in and visits their User Page (the one from Milestone 3, not Milestone 11), they should be able to edit and save the following in firebase:
  + photo
  + name
  + password
* Be sure that when a user edits their photo or name, their tweets change too such that they display the updated photo and name

**Milestone 13 - Followers**

* Think about what details this feature would need; visit social media platforms that you like and see how they implemented their follow feature
* Implement a way for users to follow one another
* Build this feature out as far as you can