Mi-12 Assignment Requirement

assignment12_category_0002

🚩: 0 [If we have any update we will mention it here]. Do check frequently to see if any updates have been made.

We're looking for a talented and eager Junior MERN Stack Developer to join our fantastic team. You'll collaborate closely with our experienced developers on state-of-the-art projects utilizing top technologies such as MongoDB for databases, Express.js for backend development, React.js for frontend interfaces, and Node.js for the runtime environment. This is your opportunity to enhance your skills and contribute to creating groundbreaking web applications that truly make a difference. Expect to work on features like user authentication, real-time updates, and responsive design, all while adhering to best practices in security and performance. If you're passionate about coding and eager to learn and grow in a dynamic environment, we want to hear from you!

Objective

You must develop a **Forum (an online platform where people can hold conversations in the form of posted messages)** using the MERN stack.

Key Rules:

- Include a minimum of 20 notable GitHub commits on the client side
- Include a minimum of 12 notable GitHub commits on the server side
- Add a meaningful readme.md file with the name of your website, Admin username, password and live site URL. Include a minimum of 10 bullet points to feature your website.

- Make it responsive for all devices. You need to make it responsive for mobile, tablet and desktop views. Make the dashboard responsive as well.
- After reloading the page of a private route, the user should not be redirected to the login page.
- Use the Environment variable to hide the Firebase config keys and Mongodb credentials.
- Don't use any Lorem ipsum text in your website.
- Show sweet alert/toast/notification for all the crud operations, successful authentication login, and sign-up. Don't use the default browser alert.
- Implement tanstack query in all the data fetching functionality (For GET method only)

Main Tasks

HomePage:

1. Navbar has a **logo+website name, Home, Membership, Notification icon,** and **Join US (when not logged in)** button. If the user is logged in, his/her profile picture should appear on the navbar.

   If the user clicks on the **profile picture,** a drop-down will appear with the following items: **User name (not clickable), Dashboard,** and **Logout** button.

2. The banner section will have a search bar. Your search word should be based on the tags you used in your posts **(See point 5).** The search implementation functionality should be done in the backend.

   Note: Do the necessary beautification of this section.

3. Create a section that will have all the tags the site offers to use in the posts. Users can use these tags to search for any post.

4. Create a section for the announcement to show all the announcements. If no announcement is made, the section will not be visible (**See point 16**). If there is an announcement, the notification icon will show the announcement count.

   Hints: Keep the announcement in a collection to get the announcement count.

5. By default, you will show all the posts on your Homepage from newest to oldest order. Each post will show the **picture of the author, post title, tags, time, comments count** and **votes count.**

   Implement a Sort by Popularity button that will show popularity based on the total vote counts (**Hints: UpVote - DownVote. Here you need to calculate the total vote, to do this you need to subtract the DownVote count from the UpVote count**). The list will be in descending order.

   Add the pagination feature on the Home page making sure that there will be 5 posts per page.

   Hint-1:

```
db.collection.aggregate([
 {
  $addFields: {
   voteDifference: { $subtract: [“$upVote”, “$downVote”] }
  }
 },
 {
  $sort: { voteDifference: -1 }
 }
])
```

   In this example:

   - **$addFields:** This stage adds a new field called voteDifference to each document, calculated as the difference between upVote and downVote.

- **$sort:** This stage sorts the documents based on the newly added voteDifference field in descending order (-1).

Adjust the collection name and field names according to your actual MongoDB setup. This aggregation pipeline calculates the difference between upVote and downVote and sorts the documents based on this difference.

Hint-2: For comments count, Create a collection for comments and make sure to store the post title along with the user comment. The comment count will be based on the post title.

6. Clicking on a post/search result, the user will be redirected to the page (**`.../post/${_id}`**) where it will show the post details. The post details will have the following:

    1) Author image
    2) Author name
    3) Post Title
    4) Post Description
    5) Tag
    6) Post Time (store post time while adding a post)
    7) Comment button
    8) UpVote icon
    9) DownVote icon
    10) Share button

    Below the post details, there will be a comment section. A user needs to log in to make a comment, give a vote**(upvote/ downvote)** and share a post.

    Explore the [react-share] package and implement it in your **Share Button** to make the posts shareable.

    Hint-1: Keep two properties **upvote** and **downvote** in your collection. By default, they will be zero. If a user clicks on upvote, the count will increase by 1. However, if the user clicks on a downvote of the same post the count will also increase by 1.

Hint-2: The share URL of your react-share will be the dynamic URL.
const shareUrl = `.../post/${_id}`;

Note: Any user **can** make more than one comment.

7. **Membership Page (Private Route) :**
   This will be the Payment page where the user has to pay N taka/dollar to become a member of this site.

   If a user becomes a member, he will receive the **Gold badge** and can do **more than 5 posts.**

User Dashboard (Private Route):

Note: **This must be a dashboard layout**

8. When a user clicks on the Dashboard, he/she will be redirected to a page where there will be the following routes:
   A. My Profile
   B. Add Post
   C. My Posts

9. **My Profile:**
   This page will have the user's name, image, email, badges, and my 3 recent posts (show necessary information).

   There will be **two** badges and these badges will be visible only on the My Profile page when the conditions are fulfilled:

   1. **Bronze Badge:** If a user registers on the site, he/ she will receive the Bronze badge.
      Or,
   2. **Gold Badge:** If a user becomes a member **(see req 7)**, he/ she will be rewarded the Gold badge.

Note: You can use a picture/icon for the badge. It is up to you. Keep it relevant.

10. **Add Post:**
    This page will have a form with the following fields:

    - Author Image
    - Author Name
    - Author Email
    - Post Title
    - Post Description
    - Tag (Select a tag from the dropdown. Use the React-select npm package. Implementing this package is optional.)
    - UpVote (By default zero)
    - DownVote (By default zero)

    Note: A normal user can add up to 5 posts. If he/she exceeds the post count, the Add Post Page will only show the **Become a Member** button with a relevant message. Hide the form. On clicking the **Become a Member**, users will be redirected to the **Membership** Page.

    Hints: When the user visits this page hit an API to get the count of posts he made from the posts collection then render this button or post form depending on the value.

11. **My Posts:**
    If a user visits this page, he/she will be able to see all the posts he/she posted. Show them in tabular form. Each row will have:

    - Post Title
    - Number of votes
    - Comment Button
    - Delete Button

12. On clicking the Comment button, the user will be redirected to a page where he/she can see **all the comments on that post (/comments/postId)**. The user can Report a comment on the Comment page. Show comments in

tabular form where each row will have the email of the commenter, the comment text, feedback, and a Report button. By default, the Report button will be disabled. The Feedback column will have a dropdown with 3 feedbacks (**these will be static and totally up to you but make sure to keep it relevant**). If a user selects a feedback reason, the Report button will become active. Once the Report button is clicked, it will be disabled. (**See point 14 to see the admin's report-related task**)

Note: The comment column will text up to 20 characters in length. If the said length exceeds, there will be ellipses with the **Read More** link. On clicking the **Read More** link, he/she will see the full comment on a modal.

13. **Join US Page:** This is the page to implement authentication. Users will see a login form. Add at least one social login in both the Join Us and Register pages. There will be a link to toggle between Register and Join Us page.

    Don't forget to give the necessary badge to the user while joining the forum for the first time. See req. 9

    Implement react-hook-form in the registration & login page.

    Note: Do not enforce the email verification method and forget & reset password method, as it will inconvenience the examiner. If you want, you can add it after receiving the assignment result.

Admin Dashboard (Private Route):

Note: **This must be a dashboard layout**

14. When an admin clicks on the Dashboard, he/she will be redirected to a page where there will be the following routes:
    A. Admin Profile (**See Bonus point 2**)
    B. Manage Users
    C. Reported Comments/Activities
    D. Make Announcement

15. **Manage Users:**

    Show all the users in a tabular form where each row will have:

    - User name
    - User email
    - Make admin
    - Subscription Status(Membership)

    The admin can make a user admin by clicking on the **Make Admin** button. Implement a server-side search functionality to find a specific user (via username).

16. **Make announcement:**

    This page will have a form with the following fields:

    - Author Image
    - Author Name
    - Title
    - Description

    What announcement you will make is entirely up to you but make sure to keep it relevant.

    The announcement will appear in the section discussed in **point 4**.

17. **Reported Activities/Comments Page:**

    a. An admin will see all the reports made in requirement 12.
    b. Show necessary information about the report/feedback to the admin.
    c. Implement necessary admin functions that you want an admin to take against these reports. This feature is completely up to you. Make the actions relevant. Think about a Facebook group admin. Write in detail about this feature in your readme file.

Challenges Tasks

1. **Admin Profile:**
   This page will have the admin's name, image, email, number of posts, number of comments, and number of users.

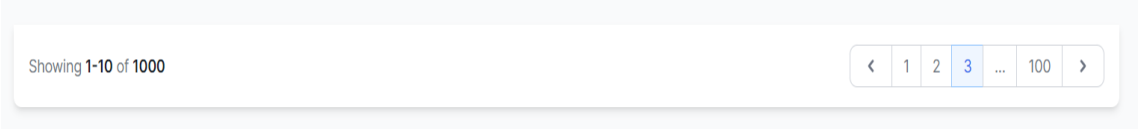
   Also, Create a pie chart to show the total number of posts, numbers of comments, and number of users of the entire site.

   Below the profile section, there will be a form where the admin can add tags. These added tags will appear in the dropdown discussed in point 10 (main task).

   Note: Create a collection to store the tags.

2. Implement JWT on login (Email/Password and social) and store the token.

3. Implement pagination at the footer of all the tables you have implemented (show 10 users at a time). For example,



Optional Tasks

You have to implement **any of the tasks** listed below:

1) If a user clicks on upvote, the count will increase by 1. However, if the user clicks on a downvote of the same post, the count will increase by 1 but the upvote **will** decrease by 1 and vice versa. This means a user can not give an upvote & downvote at the same time for the same post. **Check YouTube's like and dislike features.**

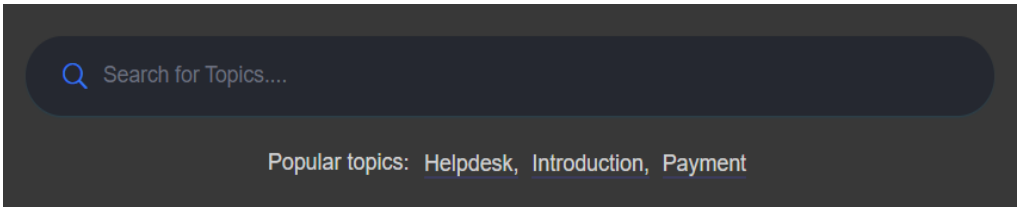
2) Implement the About Me section on the User's My Profile Page. This section will have an **Edit** button and clicking on the **Edit** button will render a form with the About Me field. On successful submission, the About Me information will appear on the About Me section of the My Profile Page.

3) Implement the react-awesome-button and React-select packages.

4) Implement a feature where the user will be able to change the visibility of his/her posts on the My Posts page. Changing the visibility to **"Only Me/Private"** will hide that particular post from other users.

5) Implement Axios interceptor.

6) The banner section will have a background picture and a search bar with 3 recent popular searches based on the time. If a user clicks on these popular search words, he/she will see the search results. See the sample image below:



hints: store your search words in a MongoDB collection

What to Submit

1. **Assignment Category/variant:**
2. **Admin email:**
3. **Admin password:**
4. **Front-end Live Site Link:**
5. **Client Side Github Repository Link:**
6. **Server Side Github Repository Link:**